A Double-Layer-Clustering Differential Evolution by Speciation and Self-Adaptive Strategies for Multimodal Optimization

Qingxue Liu
College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao, 266590, China
Department of Electrical Engineering, Tshwane University of Technology, Pretoria, 0001, South Africa
hmxue2000@163.com

Shengzhi Du
Department of Mechanical Engineering, Mechatronics, and Industrial Design, Tshwane University of Technology, Pretoria, 0001, South Africa
dushengzhi@gmail.com

Barend Jacobus van Wyk
Faculty of Engineering and Built Environment, Tshwane University of Technology, Pretoria, 0001, South Africa
vanwykb@tut.ac.za

Yanxia Sun
Department of Electrical and Electronic Engineering Science, University of Johannesburg, Johannesburg, 2006, South Africa
sunyanxia@gmail.com

Abstract
Multimodal optimization is to find and maintain as many global and local optima of a function as possible. Niching techniques based on multi-population and clustering have been proved to be effective and efficacious for tackling the multimodal optimization problems. How to enhance the diversity of the population and improve the global search ability of the algorithm to locate more optima is a big challenge. To address this objective, a Double-Layer-Clustering Differential Evolution (DE) based on Speciation (SDLCDE) and integrated it with Self-adaptive strategy (SSDLCDE) is proposed to solve the multimodal optimization problems. The first layer clustering based on speciation is used to divide the entire population into multiple subpopulations to locate the global and local optima. Then all the species seeds from each species form a subpopulation for the global search. To test the performance of our proposed algorithms, both SSDLCDE and SDLCDE are compared with 17 state-of-art niching algorithms on 29 multimodal problems with different dimensions. The Experiment results demonstrate that both the proposed algorithms outperform or perform comparably to the 17 niching algorithms on all the test functions.
Keywords: Differential evolution, multimodal optimization, niching algorithm, speciation, self-adaptive strategy.

1. Introduction

Evolutionary algorithms such as genetic algorithms, immune algorithms, colony algorithm, artificial neural network algorithm, Particle Swarm Optimization (PSO), and more recently DE have proved to an rapid and robust global techniques in solving difficult and complex optimization problems [1-7].

However, classical evolutionary algorithms were usually designed as a task of dealing with unimodal optimization problems which have only a single optimum to detect. As we all know, in the real-world, many optimization problems are multimodal problems which have multiple optimal solutions including global and local optima. Even if some optimization problems have only a single global optimum, there are many local optima in the feasible solution space, which will guide the algorithm to fall into the local optimum. Furthermore, in real-world applications, there is no need to find the global optimal solution because of physical and/or cost constraints [8, 9]. For instance, in many cases, it will increase more computational costs to get the global optimal solution. In fact, the local optimal solution has been able to meet the needs of the actual production. Hence, it is great practical significance to explore and study an optimization method for tackling multimodal optimization problems.

The niche is a concept of biology, which refers to a kind of living environment and in a specific ecosystem. In the environment, living beings always evolve with species that have the same characteristic; thereby different species form different niches. Actually the essence of niching technique is to find multiple global optima of the multimodal problems as well as local optima by forming and maintaining different niche (species) [10, 11]. Inspired by this characteristic of species, a number of techniques commonly called as “niching” methods have been proposed and developed in the past, which were specially designed to solve multimodal optimization problems. In general, niching methods are combined with a classical evolutionary algorithm, and divide a single population into multiple stable subpopulations; each subpopulation is then responsible for searching for one of multiple optima.

In evolutionary algorithm literatures, some classical examples include which include crowding [12], fitness sharing [13], deterministic crowding [14], sequential niche technique [15], clustering [16], restricted tournament selection [17], clearing [18], parallelization [19], and species [20]. In 2002, for the first time, a niching PSO method was proposed to handle multimodal optimization problems [21]. From then on a number of niching techniques were integrated with PSO to tackle multimodal optimization problems, such as fitness Euclidean distance ratio based PSO (FER) [22], species-based PSO (SPSO) [9, 23], ring topology PSO (including r3pso, r2pso, r3pso-lhc and r2pso-lhc) [2], niching PSO with local search [24], and distanced-based locally informed PSO (LIPS) [25].

The basic DE is a relative new and simple optimization technique compared with other evolutionary algorithms. Since the DE algorithm was proposed, it has been successfully applied to solve various real-world problems from the different areas of science and technology. In recent years, numerous new variants of DE algorithm have been developed using neighborhood mutation based on Euclidean distance, such as neighborhood based crowding DE (NCDE), neighborhood based speciation DE (NSDE), and neighborhood based sharing DE (NShDE) [24]. Subsequently another variant of DE with clustering partition, has

also been proposed to deal with multimodal optimization problems, which includes CCDE, CSDE, Self-CCDE, and Self-CSDE [26].

However, both neighborhood mutation and clustering partition strategies are in essence that a single population is divided into many stable subpopulations in order to detect multiple optimal or suboptimal solutions. Just like basic DE algorithm, this multi-population strategy will also suffer from the problem of falling into local optima, when it is applied to multimodal optimization problems. For instance, as the evolution process moves on, if all the individuals gather around some of the peaks, and don't cover all the peaks, it will be extremely difficult for them to escape from this area to find new peaks.

An efficient niching optimization algorithm should meet at least two requirements. First, it tries to maintain the diversity of population to locate as many optima as possible. Second, in order to cover the regions of all the peaks, it has the ability of escaping of the local optimal to search for the new peeks. To address the two above-mentioned issues, in this paper, a double-layer-clustering DE based on speciation (SDLCDE) is proposed, and the SDLCDE algorithm combined with self-adaptive parameter control strategy, named Self-adaptive SDLCDE (SSDLCDE), was also given. The details of the two methods are described in the section 3. Our algorithms have two layers structure, in the first layer, the whole population is divided into a number of independent subpopulations using speciation technique. Soon afterwards, these subpopulations located respectively the global optima and the local optima. This layer structure is responsible for locating as many optima as possible. Moreover, there is no competition among niches (subpopulations), and each niche evolves independently so that the diversity of population is maintained during the iteration of the algorithm. For the second layer structure, species seeds from each subpopulation of the first layer are formed into a new subpopulation to detect the missed optima in the entire solution space. This layer structure increases the exploration ability of the algorithm, which help to individuals escape from the local optimal.

The remainder of this paper is organized as follows. Section 2 describes the basic DE algorithm and introduces the background of the speciation and reviews the classical speciation-based niching techniques. In Section 3, the proposed algorithms SSDLCDE and SDLCDE are described and presented in sufficient details. Section 4 gives the problems definition and the corresponding experimental setup. The experiment results are discussed and presented in Section 5. Finally, Section 6 concludes this paper with pertinent observations.

2. Scientific Background and Related Works

A. Basic DE Algorithm

1) Differential Evolution

The basic DE algorithm was introduced by Storn and Price in [5, 27], which implements four main steps (including initialization, mutation, crossover, and selection operations) to update the population during the iteration. The standard DE algorithm has three key parameters: 1) population size *NP*; 2) crossover probability *CR*; scaling factor *F*.

For an optimization problem with *D* dimensions, the *i*th individual x_i (so-called target vector)

in the population of DE is represented as

$$x_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,D}] \qquad i = 1, 2, \ldots, NP \qquad (1)$$

Where, *NP* is the population size.

2) Mutation Operation

At each generation *G*, DE implements the mutation operation to produce a mutant vector v_i by a mutation strategy. Some mutation strategies used in recent years are given as below:

"DE/rand/1"[28]: $v_i = x_{r1} + F(x_{r2} - x_{r3})$ (2)

"DE/rand/2"[29]: $v_i = x_{r1} + F(x_{r2} - x_{r3}) + F(x_{r4} - x_{r5})$ (3)

"DE/best/1"[28]: $v_i = x_{best} + F(x_{r1} - x_{r2})$ (4)

"DE/best/2"[28]: $v_i = x_{best} + F(x_{r1} - x_{r2}) + F(x_{r3} - x_{r4})$ (5)

"DE/current-to-rand/1(without crossover)"[30, 31]:

$$u_i = x_i + K(x_{r1} - x_i) + F(x_{r2} - x_{r3}) \qquad (6)$$

"DE/current-to-best/1"[28, 32, 33]:

$$v_i = x_i + F(x_{best} - x_i) + F(x_{r1} - x_{r2}) \qquad (7)$$

"DE/current-to-best/2"[33]:

$$v_i = x_i + F(x_{best} - x_i) + F(x_{r1} - x_{r2}) + F(x_{r3} - x_{r4}) \qquad (8)$$

"DE/current-to-*p*best/1(without archive)"[34]:

$$v_i = x_i + F(x_{pbest} - x_i) + F(x_{r1} - x_{r2}) \qquad (9)$$

"DE/current-to-*p*best/1(with archive)"[34, 35]:

$$v_i = x_i + F(x_{pbest} - x_i) + F(x_{r1} - \tilde{x}_{r2}) \qquad (10)$$

"DE/rand-to-best/1"[28, 33]: $v_i = x_{r1} + F(x_{best} - x_{r1}) + F(x_{r2} - x_{r3})$ (11)

"DE/rand-to-best/2"[33]:

$$v_i = x_{r1} + F(x_{best} - x_{r1}) + F(x_{r2} - x_{r3}) + F(x_{r4} - x_{r5}) \qquad (12)$$

In the equations (2) – (12), the indices $r_1 - r_5$ are generated randomly once for each mutant vector within the range [1, *NP*]. Furthermore, they must be mutually exclusive and different from the index *i*. *F* is the scaling factor which is a positive control parameter for scaling the difference vector. *K* in Equation (5) is randomly chosen within the range [0, 1]. x_{best} is the best individual with the best fitness value in the population. x_{pbest} in equation (9) and (10) is randomly chosen as one of the top 100*p*% individuals in the current population with *p* ranging on [0, 1].

3) Crossover Operation

After the mutation phase, using each pair of the target parent vector x_i and its corresponding mutant vector v_i, crossover operation is employed to generate a trial vector $u_{i,j}$. For the basic DE algorithm, DE implements the binomial (uniform) crossover operation defined as follows [5]:

$$u_{i,j} = \begin{cases} v_{i,j} \ , \text{ if } (rand_{i,j} \le CR) \text{ or } (j = j_{rand}) \\ x_{i,j} \ , \text{ otherwise} \end{cases} \tag{13}$$

Where, $i = 1, 2, \ldots, NP$, $j = 1, 2, \ldots, D$, j_{rand} is a randomly chosen natural number in the range [0, D], CR is the crossover probability, which is a user-specified control parameter of DE, and $rand_{i,j}$ is a uniform random number in the range [0, 1]. In fact, owing to the introducing of $rand_{i,j}$, it is guaranteed that the trial vector $u_{i,j}$ differ from its target vector x_i.

4) Selection Operation

At the last stage, the selection operation is performed by comparing the target parent vector x_i with the corresponding trial vector u_i. The one with the best function value is selected to the next generation. For the maximization problem, the selected vector is given by

$$x_i^{G+1} = \begin{cases} u_i^G \ , \text{ if } f(u_i^G) \ge f(x_i^G) \\ x_i^G \ , \text{ otherwise} \end{cases} \tag{14}$$

B. Speciation

Similar to niching algorithms, speciation is essentially a centroid-based clustering technique by which a whole population is divided into multiple different species centered a species seeds [20, 21, 36, 37]. Whereas speciation doesn't strictly belong to the classical clustering method, such as density-based method, hierarchical clustering method, and k-means method, it simplifies the clustering process elitist-based niching though designating the best-fit individual of the remaining population to as species seed [38]. In 2002, a species conserving genetic algorithm was proposed to solve multimodal optimization problems and has proved to be very effective [20]. Subsequently, a species-based PSO was proposed, and the experiments showed that it is very effective in handling multimodal optimization functions with lower dimensions [8, 9]. However, in order to achieve a good performance, the value of the niching radius must be specified which need to depend on the prior knowledge of the researchers. With respect to DE algorithm, species-based DE (SDE) is a classical DE algorithm proposed in 2005 [39], and its process is summarized in Algorithm 1. Based on the SDE, another two DE algorithm for multimodal problems were developed which are neighborhood-based SDE (NSDE) [24] and cluster-based SDE with self-adaptive strategy (Self-CSDE) [26], respectively. The pseudocodes of both methods are presented in Algorithm 2 and 3.

Algorithm 1 Species-based DE (DE)

Step 1 Randomly generate *NP* number of initial trial solutions.
Step 2 Evaluate all individuals in the population.
Step 3 Sort all individuals in descending order of their fitness values.
Step 4 Determine the species seeds for the current population: the most-fit individual will be set as the first species seed. Then all individuals are checked in turn from the most-fit to the least-fit against the species seeds found so far. If an individual does not fall in the radius of any seeds, it will be identified as next species seed.
Step 5 For each species, execute the basic DE algorithm:
5.1 If a species has less than *m* individuals, then randomly generate new individuals within the radius of the species seed until there are *m* individuals in the species.
5.2 If the offspring's fitness is the same as its species seed, replace this offspring with a randomly generated new individual.
End For
Step 6 Stop if a termination criterion is satisfied. Otherwise go to Step 3.

Algorithm 2 Neighborhood-based SDE (NSDE)

Step 1 Randomly generate *NP* number of initial trial solutions.
Step 2 Evaluate all individuals in the population.
Step 3 Sort all individuals in descending order of their fitness values.
Step 4 While the sorted population is not empty
Determine the species seed which is the most-fit unprocessed individual. Find the parametrically most similar (in Euclidean distance) *m* individuals of the species seed and set them as one species. Delete the processed individuals for the current population.
End While
Step 5 For each species, execute a basic DE algorithm:
If the offspring's fitness is the same as its species seed, replace this offspring with a randomly generated new individual.
End For
Step6 Keep only the *NP* most-fit (function value) individuals from the combined population.
Step 7 Stop if a termination criterion is satisfied. Otherwise go to Step 4.

Algorithm 3 Custer-based SDE with self-adaptive strategy (Self-CSDE)

Step 1 Set Crm = 0.5 and initialize a population P of NP individuals in the search space randomly.
Step 2 Evaluate all individuals in the population.
Step 3 Sort all individuals in descending order of their fitness values.
Step 4 For i = 1 to NP/M
Determine the species seed which is the most-fit unprocessed individual. Combine M-1 individuals of the population P, which are nearest to the species seed, with the species seed to form subpopulation $subpop_i$. Delete these M individuals from the current populations.
End For
Step 5 Set $S_{Cr} = \phi$.
For each subpopulation $subpop_i$
For each individual in $subpop_i$
4.1 Pick $r1$, $r2$, $r3$ from the $subpop_i$ and generate mutation vector using Equation (2).
4.2 Generate the crossover probability and use the crossover operation of DE to produce the trial vector.
4.3 Evaluate offspring u_j using the fitness function.
4.4 Compare the fitness of u_j with the most similar individual x_s (in Euclidean distance) in $subpop_i$. If $f(u_j) \geq f(x_s)$, $x_s = u_j$, save the respective crossover probability in S_{Cr}
End For
End For
Step 6 Update $Crm = mean(S_{Cr})$.
Step 7 Stop if a termination criterion is satisfied. Otherwise go Step 3.

3. Proposed algorithms

A. Motivation

The main objective of a multimodal optimizer is to detect as many peaks as possible and to ensure that they are not lost in the case of a limited population size. To address such issues, numerous techniques have been developed by neighborhood mutation strategy or speciation, or clustering, or arithmetic recombination such as NCDE [24], NSDE[24], NShDE [24], Speciation-based DE [9], Self-CCDE [26], Self-CSDE [26], and EARSDE [38]. However, as the evolution process moves on, once all the members gather around some of the peaks, it is extremely difficult for them to escape from this area, and to find new peaks. In order to overcome this problem, a Double-Layer-Clustering Differential Evolution by Means of Speciation (SDLCDE) was proposed, and the SDLCDE algorithm integrated with Self-adaptive parameter control strategy (SSDLCDE) was also proposed in this paper. The details of the two algorithms are described in the following subsections.

B. Double-Layer-Clustering Strategy

For the basic DE version [31, 32, 34, 40-47], it is designed to deal with unimodal problems which have single global optimum in the entire solution space. In this version, the mutation operation is performed between randomly picked individuals from the whole population. This will let any two particles to generate the different vector in DE/rand/1, even if the two particles are far apart from each other. However, in the final search stage of evolution, all members in general cluster around the global optimum, which result in the diversity loss of the population. So the global version can only solve a unimodal problem with single global solution, and it can't deal with a multimodal problem, efficiently.

Hence, when solving a multimodal problem, all the optima need to be detected simultaneously. Moreover, numerous subpopulations need to distribute and maintain around different peaks. In recent years, some researchers used the improved DE technique to tackle multimodal problems by multi-population strategy, such as CDE [48], species-based DE [9], NCDE [24], NSDE [24], NShDE [24], CCDE [26], CSDE [26], Self-CCDE [26], Self-CSDE [26], ARSDE [38], and EARSDE [38]. The multi-population method is that the entire population is divided into multiple subpopulations by clustering, and then each subpopulation is responsible for locating different optimal regions. As the evolution process moves on, all the individuals evolve to some optimal point. However, just like the global version, the multi-population strategy also has the diversity loss problem in the later stage of evolution, especially for a function in which the space distance between different peeks is large. The general clustering mechanism can't enable any of the optima to be found as the difference vectors must be generated using individuals from different optimal regions with relatively large magnitudes. Under these circumstances, it is very difficult for these individuals to run away from the region to search for the rest optima. Therefore, how to locate as many peaks as possible; meanwhile, maintain the diversity of the population so that the rest optimal regions can be covered by some members, is a challenging topic.

In order to address this issue, the Double-Layer-Clustering Strategy was proposed. In this strategy, we adopted double layers search mechanism. For the first layer clustering, inspired by the partition scheme of SPSO algorithm [9], a species-based clustering partition was used to divide the whole population into many subpopulations with clustering partition. Each subpopulation has M individuals which are used to generate M-1 (except the species seed) offspring by basci DE algorithm. As the iteration goes on, each subpopulation will move toward the nearby peak.

As mentioned above, there is one individual (namely species seed) in every subpopulation, which does not employ DE operation. For the second layer clustering, these species seeds are formed a new subpopulation to perform global search by DE. The second layer search can help to detect the missing peaks which were not found by the first layer clustering. To further illustrate the operation mechanism of the second layer clustering, Fig.1 is presented. As can be seen, $S_1 - S_4$ are species seeds of the first layer subpopulations, $O_1 - O_5$ are the optima to be detected, and O_5 was not located by the first layer subpopulations. Since the subpopulation consisted of species seeds runs the global DE algorithm, it is likely to find the

missing peak O_5.

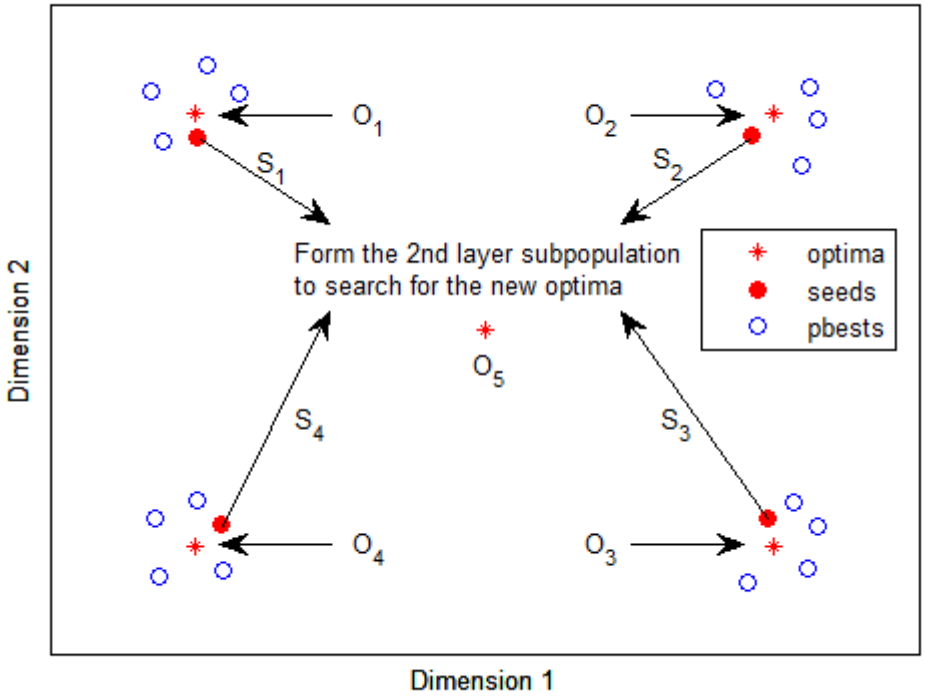

Fig.1. Illustration of operation mechanism of Double-Layer-Clustering strategy.

C. Self-Adaptive Strategy

DE algorithm has two important parameters [27, 49], one is the crossover probability (*CR*), the other is scaling factor (*F)*. They play a significance role in the performance of algorithm. A number of effective adaptive or self-adaptive parameter control approaches have been proposed [26, 31, 34, 35], which help to improve search ability of DE algorithm. According to this recommendation, a self-adaptive DE strategy was developed which is similar to the adaptation strategy adopted in the literature [34].

At each generation *G*, let CR_i be the crossover probability of every individual x_i, it is generated according to the following normal distribution.

$$CR_i = \text{randn}_i(\mu CR, 0.1) \tag{15}$$

where, μCR is the mean value and 0.1 is the standard deviation value of CR_i. Note that CR_i must be truncated to [0, 1] if necessary after calculation. Denote S_{CR} as the set of any CR_i that generates the improved solutions at generate *G*. The mean μCR is initialized to be 0.5 which is updated at the end of each generation as

$$\mu CR = (1-c)\cdot \mu CR + c\cdot \text{mean}(S_{CR}) \tag{16}$$

where, *c* is a positive constant between 0 and 1 and “mean” is a function expression which calculates the usual arithmetic mean value of the set of S_{CR}.

Similarly, at each generation G, the scaling factor F_i of each individual x_i is independently updated according to Cauchy distribution which is as follows:

$$F_i = \text{randc}_i(\mu F, 0.1) \tag{17}$$

where, μF is the location parameter and 0.1 is the scale parameter of the Cauchy distribution. Note that F_i must be truncated to the interval of [0, 1], it will be regenerated if it is more than 1 or less than 0 after the update. Denote S_F as the set of any F_i that generates the successful solutions at generation G. The initial value of the parameter μF is set to 0.5 which is updated at the end of each generation as

$$\mu F = (1-c)\cdot \mu F + c \cdot \text{mean}(S_F) \tag{18}$$

where, c and "mean" are the same to Equation (16)

D. Algorithm Framework

It is necessary to emphasize that in this paper, we intend to use the species-based Double-Layer-Clustering technique and the self-adaptive strategy to improve the performance of the proposed algorithm. First, M subpopulations are generated according to the species-based clustering partition strategy. For each species, execute a global DE variant. Second, M species seeds from M subpopulations are formed into one subpopulation, and then a global DE variant is also perform in this subpopulation. The steps of SSDLCDE were presented Algorithm 4. Note that the greedy selection scheme was employed in both the first and the second layer clustering.

E. Complexity Analysis

As we all know, the complexities of the basic DE and CDE are $O(D \bullet NP)$ and $O(D \bullet NP^2)$, respectively. In SSDLCDE algorithm, compared to CDE, there are actually two more operators than it, one is clustering partition strategy, and the other is the self-adaptive strategy. The computational complexity of clustering partition in the first layer clustering is $O(D \bullet NP^2)$, depending on which, the computational complexity of clustering partition in the second layer clustering is also naturally obtained. Since the clustering partition in the second layer clustering does not occupy computational resources, the total computational complexity is still $O(D \bullet NP^2)$. Therefore, added the complexity $O(D \bullet N^2)$ of the self-adaptive and the complexity $O(D \bullet NP^2)$ of CDE, the computational complexity of SSDLCDE algorithm remains to be $O(D \bullet NP^2)$.

Algorithm 4 The proposed SSDLCDE

Step 1 Randomly generate *NP* number of initial trial solutions x_i in the search space. Where, $i = 1, 2, \ldots, NP$.

Step 2 Evaluate all individuals in the population.

Step 3 Set $\mu CR = 0.5$, $\mu F = 0.5$, $c = 0.1$, $S_{CR} = \Phi$, $S_F = \Phi$.

Step 4 Sort all individuals into the population *sortpop* in descending order of their fitness values.

Step 5 For $j = 1$ to NP/M

5.1 Determine the species seed $\overline{x}_j$ which is the best (fitness value) unprocessed individual.

5.2 Combine M-1 individuals of the population *sortpop*, which are nearest (in Euclidean distance) to the species seed $\overline{x}_j$, with $\overline{x}_j$ to form the subpopulation $subpop_j$. And then remove these M individuals from the current population *sortpop*.

5.3 For each individual (except for x_s) in $subtpop_j$.

5.3.1 Pick $r1$, $r2$, $r3$ from the $subtpop_j$ and generate mutation vector using equation (2).

5.3.2 Generate the crossover probability and the scaling factor, and then use the crossover operation of DE to produce the trial vector.

End For

End For (Note that at this time, a total of NP-M offspring were generated.)

Step 6 For each species seed $\overline{x}_j$, which form a new subpopulation.

6.1 Pick $r1$, $r2$, $r3$ from the new subpopulation and generate mutation vector using equation (2).

6.2 Generate the crossover probability and the scaling factor, and then use the crossover operation of DE to produce the trial vector.

End For

Step 7 For $i = 1$ to NP

7.1 Evaluate the offspring u_i using the fitness function.

7.2 Compare the fitness of u_i with the most similar individual (in Euclidean distance) in population. if u_i has a better fitness, replace the most similar individual with u_i, and save the crossover probability and the scaling factor in S_{CR} and S_F, respectively.

End For

Step 8 Update $\mu CR = mean(S_{CR})$, $\mu F = mean(S_F)$.

Step 9 Stop if a termination criterion is satisfied. Otherwise go to Step 2.

4. Experimental Setups

A. Compared Algorithms

To determine the advantages of our algorithms, SSCDE and SCDE were compared with multiple state-of-the-art niching algorithms. In total, 19 different multimodal techniques are considered in our experiments, which are frequently cited in many literatures.

1) SSCDE:
2) SCDE:
3) CDE [48]: the original crowding DE.
4) SDE [39]: the original speciation-based DE.
5) CCDE [26]: the crowding DE with the clustering partition alone.
6) CSDE [26]: the speciation DE with the clustering partition along.
7) Self-CSDE [26]: the speciation DE with the clustering partition and self-adaptive strategy.
8) FER-PSO [22]: fitness-Euclidean distance ratio PSO.
9) SPSO [9]: species-based PSO.
10) r2PSO [2]: a *lbest* PSO with a topology, each member interacts with only its immediate member to its right.
11) r3pso [2]: a *lbest* PSO with a ring topology, each member interacts with its immediate member on its left and right.
12) r2pso-hlc [2]: the same as r2pso, but with no overlapping neighborhoods.
13) r3pso-hlc [2]: the same as r3pso, but with no overlapping neighborhoods.
14) CMA [50]: niching covariance matrix adaptation evolution strategy.
15) SCMA [50]: CMA with self-adaptive niche radius.
16) NShDE [24]: the neighborhood based sharing DE.
17) NSDE [24]: the neighborhood based speciation DE.
18) LIPS [25]: a distance-based locally informed PSO.
19) IWO- δ -GSO [51]:

All the algorithms were implemented using MATLAB r2014a and executed on the computer with an Intel(R) Core(TM) i5-5200(2.2GHz) and 8GB RAM.

The DE parameters were the same as those adopted in [24], while the other parameters used in this paper were adopted from their respective literatures. In our proposed algorithm, the subpopulation size M in the first layer clustering was set to be 5 and 10 in the cases of $NP \leq 200$ and $NP > 200$, respectively.

B. Numerical Benchmarks

To evaluate the performance of the proposed methods, 14 classical multimodal functions [2], and 15 scalable composition functions [52]were used. These multimodal functions were widely adopted in different published papers. As these functions have numerous global and local optima, it is difficult to track the peaks. Especially for the composition, their global optima don't always reside in the center of the search space, because they are comprised of multiple traditional functions through rotation, scaling and other complex transformations.

In our experiments, these test functions were divided into two parts: the first 14 general functions are test function set 1, the last 15 composition functions are test function set 2. The basic description and parameters setting of function set 1 and function set 2 are given in Tables 1 and 2, respectively.

For functions F1, F2, F3, F5, F7 and F10, the objective is to track the global optima and local optima, while for the rest the target is to locate the global optima and to escape the local optima. Note that all test functions are considered for maximization, hence, when the definitions are given for minimization, the functions are reversed.

Table 1 Test functions

Test Function Set 1		Test Function Set 1	
Test Function Name / Dimensions	Number of Global/Local Peaks	Test Function Name / Dimensions	Number of Global Peaks
F1: Two-Peak Trap [53]/1D	1/1	CF1: /10D and 30D	8
F2: Central Two-Peak Trap [53] /1D	1/1	CF2: /10D and 30D	6
F3: Five-Uneven-Peak Trap [20] /1D	2/3	CF3: /10D and 30D	6
F4: Equal Maxima [54] /1D	5/0	CF4: /10D and 30D	6
F5: Decreasing Maxima [54] /1D	1/4	CF5: /10D and 30D	6
F6: Uneven Maxima [54] /1D	5/0	CF6: /10D and 30D	6
F7: Uneven Decreasing Maxima [54] /1D	1/4	CF7: /10D and 30D	6
F8: Himmelblau’s function [54] /2D	4/0	CF8: /10D and 30D	6
F9: Six-Hump Camel Back [55] /2D	2/2	CF9: /10D and 30D	6
F10: Shekel’s foxholes [56] /2D	1/24	CF10: /10D and 30D	6
F11: 2-D inverted Shubert function[20] /2D	18/many	CF11: /10D and 30D	8
F12: 1-D inverted Vincent function[57] /1D	6/0	CF12: /10D and 30D	8
F13: 2-D inverted Vincent function[57] /2D	36/0	CF13: /10D and 30D	10
F14: 3-D inverted Vincent function[57] /3D	216/0	CF14: /10D and 30D	10
		CF15: /10D and 30D	10

Table 2. Parameters and criteria for the 15 test functions conditions

Function No.	ε	r	Population size	Number of function ealuations
F1	0.05	0.5	50	10,000
F2	0.05	0.5	50	10,000
F3	0.05	0.5	50	10,000
F4	0.000001	0.01	50	10,000
F5	0.000001	0.01	50	10,000
F6	0.000001	0.01	50	10,000
F7	0.000001	0.01	50	10,000
F8	0.0005	0.5	50	10,000
F9	0.000001	0.5	50	10,000
F10	0.00001	0.5	50	10,000
F11	0.05	0.5	250	100,000
F12	0.0001	0.2	100	20,000
F13	0.001	0.2	500	200,000
F14	0.001	0.2	1000	400,000
CF1-CF15(10D)	0.5	1	600	300,000
CF1-CF15(10D)	1	1	1000	800,000

C. Population Size and Maximal Number of Evaluations

In our experiment, a level of accuracy ε (typically $0< \varepsilon <1$) needs to be specified to compare different techniques fairly. This parameter is used to indicate how close the fitness values of the computed solutions to the known global and local peaks are. That is say that if the absolute value from a computed solution to a known global or local optimum is below ε , then the peak is considered to have been found. The level of accuracy (ε), niching radius (*r*), population size (*NP*), and maximal number of function evaluations allowed are listed in Table 2. Note that these parameter settings were applied to all compared algorithms. In general, different population size and different maximum numbers of functions evaluations are determined by complexity degrees of test functions and the number of their peaks. So a function with a large number of peaks requires a larger population size and more function evaluations.

D. Performance Criteria

1) Success Rate

Success Rate is an important parameter to assess the performance of the algorithm. It is the percentage of total independent runs in which all global and/or local peaks are successfully detected within the budget of maximum number of function evaluations. Note that success rate must depend on the level of accuracy mentioned in the previous subsection. Level of accuracy indicates varying degree of proximity to the know peaks. Note also, that for complex test functions, if a higher level of accuracy is selected, all peaks may not be found. In this case, the corresponding success rate is zero.

2) Average number of optima found [58].

Average number of optima found denotes the average number of peaks located by an algorithm within the budget of maximum number of function evaluations. To compare the performance of different multimodal algorithms, all performances are calculated and averaged over 30 independent runs on each test function. In order to determine the statistical significance of the advantage of our methods, *t*-test and the Mann-Whitney-Wilcoxon rank sum test on the average numbers of peaks detected by two compared techniques at 5% significance level are applied [26, 51], which are presented as "*t*-test / Wilcoxon test" in the second row of each cell. The marks "1" and "0" indicate that the best algorithm is statistically better than or equals to the compared algorithms.

5. Results and discussion

This section gives and discusses the experimental results analyses of our comparative study. On all the benchmark problems, all algorithms were executed until all known peaks were found or the maximum number of function evaluations was exhausted. Our two methods were compared with other state-of-the-art algorithms which were recorded and presented in Tables 4-12.

A. Success Rate

The success rate is an important indicator to for evaluating an optimization algorithm, which reflects the reliability of an optimization method. Table 3 shows the results of test function set 1. The ranks of each algorithm are given in parentheses while the total ranks (summation of all the individual ranks) are presented in the last row of this table. It can be seen from Table 3 that the proposed two methods achieve much higher success rate than other optimization algorithms on these 14 test functions. Our proposed algorithms rank on top two among all the compared algorithms. Moreover, if we focus on observing the prosed two algorithms, SSDLCDE outperforms SDLCDE on all the test functions, which shows the effectiveness of the proposed self-adaptive strategy. Comparing these two algorithms with both the original SDE and its cluster-based SDE (CSDE), we can also see that the proposed algorithms have better performance than in all the cases. Especially for SSDLCDE algorithm, it achieved 100% success rate on all 14 test functions except for the last two.

Note that the results for the test functions set2 (CF1-CF15) are not reported, because these

composition functions are much more complex as opposed to the ordinary functions, they have more complicated fitness landscapes and irregular peak distributions, no algorithm is able to get a nonzero success rate. For this reason, “average number of optima found” is also used as another valuation indicator in the next subsection.

Table 3. Success rates (%) for test function set 1 (F1-F14) and the respective ranks (in parentheses)

Fun	SSDL CDE	SDL CDE	CDE	SDE	CCDE	CSDE	FER-PSO	SPSO	r2pso	r3pso	CMA	SCMA
F1	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	82(10)	44(12)	76(11)	84(9)	100(1)	100(1)
F2	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	44(12)	88(11)	96(10)	100(1)	100(1)
F3	100(1)	100(1)	100(1)	96(6.5)	100(1)	100(1)	20(9)	4(12)	8(10.5)	8(10.5)	96(6.5)	92(8)
F4	100(1)	100(1)	28(10)	72(9)	100(1)	100(1)	84(8)	88(6.5)	92(5)	88(6.5)	20(11)	12(12)
F5	100(1)	100(1)	72(12)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)
F6	100(1)	100(1)	28(10)	60(9)	100(1)	86(7)	100(1)	92(5)	88(6)	72(8)	8(11)	0(12)
F7	100(1)	100(1)	60(12)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)	100(1)
F8	100(1)	90(2)	0(11.5)	72(5)	72(5)	88(3)	72(5)	0(11.5)	28(9)	24(10)	68(8)	70(7)
F9	100(1)	100(1)	0(11.5)	100(1)	90(7)	100(1)	96(6)	0(11.5)	56(10)	60(9)	100(1)	66(8)
F10	100(1)	100(1)	52(9)	32(10)	92(5)	100(1)	100(1)	56(8)	88(6)	76(7)	18(11)	4(12)
F11	100(1)	100(1)	72(6)	46(9)	100(1)	100(1)	52(8)	0(12)	4(10.5)	4(10.5)	100(1)	60(7)
F12	100(1)	100(1)	56(10.5)	48(12)	88(3)	68(5.5)	60(7.5)	72(4)	68(5.5)	56(10.5)	58(9)	60(7.5)
F13	94(1)	90(2)	8(5)	0(9)	86(3)	30(4)	0(9)	0(9)	0(9)	0(9)	0(9)	0(9)
F14	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
Total ranks	14	16	101.5	75.5	32	29.5	68.5	106.5	96.5	103	72.5	87.5

B. Average Number of Optima Found

The average number of optima found is another important criterion for comparing different niching techniques. As mentioned in the previous subsection, for a complex or difficult problem, if not all the optima are detected within each independent run, success rate can be zero. Under these circumstances, in order to further clarify the comparative results of various niching algorithms, the average number of optima found for each test function is used in this paper. Tables 4-6 respectively list the comparative results of various niching algorithms on

test function set 1 and 2 in terms of the average number of global optima found.

1) Results on test function set 1

Table 4 shows the results of 12 niching algorithms on test function set 1 (F1-F14). Please note, beside the usage of ranks, *t*-test and Mann-Whitney-Wilcoxon rank are also executed here to compare our two with other competitive techniques, which is able to ensure statistical significance in the advantage of our algorithms over other comparative algorithms. For instance, the cell in the sixth row (i.e., F6) and the fourth column (i.e., SDE) shows that the average number of global optima is 4.6 over 30 independent runs. The number "9" in the parentheses denotes that SDE ranks the ninth among the 12 competitive algorithms (i.e., behind SSDLCDE, SDLCDE, CCDE, CSDE, FER-PSO, SPSO, r2pso and r3pso in that order), while the mark "1/1" indicates that the best algorithm (SSDLCDE) significantly outperforms SDE statistically in terms of *t*-test and Wilcoxon test, respectively. From this table, it can be seen that both of the proposed algorithms have located all the global peaks on each of the first 12 test functions. For the last two test functions, although all the niching algorithms don't obtain 100% success rate, it is observed that our proposed algorithms are able to locate fur more global peaks that other competitors.

2) Results on test function set 2

All 15 functions in test function set 2 are composition functions with much more complicated fitness landscapes as opposed to the functions in test set 1. The average numbers of global optima located by each niching technique on these 15 composition functions with 10 and 30 dimensions are presented in Tables 5 and 6, respectively. From these two tables, we can observe that SSDLCDE and SDLCDE are significantly either better than or equivalent to other niching algorithms on all 15 composition functions. Especially in the case of high dimensions (i.e., 30D), our proposed two algorithms show overwhelming advantages over other niching algorithms on most of test functions in terms of average global optima detection.

C. Further Comparison with Other Niching Algorithms

In order to demonstrate the advantages of our proposed algorithms, we have also chosen seven other classical algorithms for comparison on test functions set 1 and 2. The seven niching algorithm are Self-CSDE [26], NShDE [24], NSDE [24], LIPS [25], r2pso-lhc [2], r3pso-lhc [2] and IWO- δ -GSO [51], respectively. Tables 7 and 8 list the comparative results on all 29 test functions. Note that the results listed of the last niching algorithm are derived directly from its corresponding reference [51]. Since function 11 in test set 1 andcomposition functions 10-15 in test set 2 were not considered in this reference, the results with respect these test functions are represented by "NA". From the results of Tables 7 and 8, it is clear that both proposed algorithms work well and show the superior performance than other seven compared niching techniques on most of test functions. In order to demonstrate the significant differences between the proposed algorithm and each of seven compared niching algorithms, *t*-test and Wilcoxon test on the average numbers of optima found by two competitors are also conducted here. We can see from these Tables 7 and 8 that the proposed

methods have significant advantages than the other algorithms in terms of overall performance.

Table 4. Average number of global peaks found for test function set 1 (F1-F14) and the respective ranks (in parentheses), *t*-test and Wilcoxon test on the average number of peaks for test functions are shown in the second row of each cell, which is presented as "*t*-test / Wilcoxon test".

Fun	SSDL CDE	SDL CDE	CDE	SDE	CCDE	CSDE	FER-PSO	SPSO	r2pso	r3pso	CMA	SCMA
F1	1(1) NA	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	0.72(11) 1/1	0.48(12) 1/1	0.76(10) 1/1	0.84(9) 1/1	1(1) 0/0	1(1) 0/0
F2	1(1) NA	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	0.44(12) 1/1	0.88(11) 1/1	0.96(10) 1/1	1(1) 0/0	1(1) 0/0
F3	2(1) NA	2(1) 0/0	2(1) 0/0	1.96(6) 0/0	2(1) 0/0	2(1) 0/0	0.8(9) 1/1	0.24(12) 1/1	0.48(11) 1/1	0.6(10) 1/1	1.95(7) 0/0	1.92(8) 1/1
F4	5(1) NA	5(1) 1/1	3.84(10) 1/1	4.72(8) 1/1	5(1) 0/0	5(1) 0/0	4.84(6) 0/0	4.88(5) 0/0	4.68(9) 0/0	4.74(7) 0/0	0.6(11) 1/1	0.04(12) 1/1
F5	1(1) NA	1(1) 0/0	0.72(12) 1/1	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0
F6	5(1) NA	5(1) 0/0	3.96(10) 1/1	4.6(9) 1/1	5(1) 0/0	4.8(7) 0/0	5(1) 0/0	4.92(5) 0/0	4.88(6) 0/0	4.72(8) 1/1	0.64(11) 1/1	0(12) 1/1
F7	1(1) NA	1(1) 0/0	0.6(12) 1/1	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	0.96(11) 0/0
F8	4(1) NA	4(1) 0/0	0.32(12) 1/1	3.72(4) 1/1	3.56(6) 1/1	3.8(3) 1/0	3.68(5) 1/1	0.84(11) 1/1	2.92(9) 1/1	2.76(10) 1/1	3.43(8) 1/1	3.44(7) 1/1
F9	2(1) NA	2(1) 0/0	0.04(12) 1/1	2(1) 0/0	1.86(7) 1/1	2(1) 0/0	1.96(6) 0/0	0.08(11) 1/1	1.44(10) 1/1	1.56(9) 1/1	2(1) 0/0	1.6(8) 1/1
F10	1(1) NA	1(1) 0/0	0.52(9) 1/1	0.32(10) 1/1	0.92(5) 1/1	1(1) 0/0	1(1) 0/0	0.56(8) 1/1	0.88(6) 1/1	0.76(7) 1/1	0.18(11) 1/1	0.04(12) 1/1
F11	18(1) NA	18(1) 0/0	17.7(5) 0/0	12.4(9) 1/1	18(1) 0/0	18(1) 0/0	17.4(6) 0/0	8.52(12) 1/1	15.2(8) 1/1	15.6(7) 1/1	12(11) 1/1	12.04(10) 1/1
F12	6(1) NA	6(1) 0/0	5.56(7.5) 1/1	4.88(12) 1/1	5.9(3) 0/0	5.8(5) 0/0	5.36(10) 1/1	5.6(6) 1/1	5.52(9) 1/1	5.16(11) 1/1	5.81(4) 0/0	5.56(7.5) 1/1
F13	35.98(1) NA	35.96(2) 0/0	33.8(4) 1/1	22.8(10) 1/1	34.6(3) 0/0	30.2(5) 1/1	23.6(9) 1/1	25.7(6) 1/1	21.8(12) 1/1	22.2(11) 1/1	23.7(8) 1/1	24.6(7) 1/1
F14	192(1) NA	190.4(2) 0/0	152(4) 1/1	50.6(8) 1/1	160.12(3) 1/1	102.5(5) 1/1	68.6(7) 1/1	70.1(6) 1/1	40.6(10) 1/1	45.4(9) 1/1	32.5(11) 1/1	0.6(12) 1/1
Total ranks	14	16	100.5	81	35	34	74	108	113	110	87	109.5

Table 5. Average number of global peaks found for test function set 2 (CF1-CF15) with D=10 and the respective ranks (in parentheses) , *t*-test and Wilcoxon test on the average number of peaks for test functions are shown in the second row of each cell, which is presented as "*t*-test / Wilcoxon test".

Fun	SSDL CDE	SDL CDE	CDE	SDE	CCDE	CSDE	FER-PSO	SPSO	r2pso	r3pso	CMA	SCMA

CF1	6.91(1) NA	6.88(2) 0/0	0(10.5) 1/1	1.79(6) 1/1	5.1(4) 1/1	6.6(3) 0/0	1.08(7) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1(8) 1/1	2(5) 1/1
CF2	4.1(1) NA	4(2.5) 0/0	1.2(9) 1/1	1.3(8) 1/1	3.7(4) 1/1	4(2.5) 0/0	2(5) 1/1	0(11) 1/1	0(11) 1/1	0(11) 1/1	1.4(7) 1/1	1.52(6) 1/1
CF3	6(1) NA	5.96(4) 0/0	0.7(9) 1/1	1.5(8) 1/1	6(1) 0/0	6(1) 0/0	2.5(5) 1/1	0(11) 1/1	0(11) 1/1	0(11) 1/1	2.08(7) 1/1	2.21(6) 1/1
CF4	5.7(1) NA	5.6(2) 0/0	0(9.5) 1/1	0(9.5) 1/1	4.7(4) 1/1	5.42(3) 1/1	0(9.5) 1/1	0(9.5) 1/1	0(9.5) 1/1	0(9.5) 1/1	0.4(5) 1/1	0.2(6) 1/1
CF5	6(1) NA	6(1) 0/0	1.1(8) 1/1	1.3(6) 1/1	5.3(4) 1/1	6(1) 0/0	2(5) 1/1	0(11) 1/1	0(11) 1/1	0(11) 1/1	1(9) 1/1	1.2(7) 1/1
CF6	3.83(1) NA	3.71(2) 1/1	0(10.5) 1/1	1.4(7) 1/1	3(3.5) 1/1	3(3.5) 1/1	1.2(8) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1.54(6) 1/1	2.6(5) 1/1
CF7	3.37(1) NA	3.23(2) 0/0	0(10.5) 1/1	1(7) 1/1	1.8(4) 1/1	1.86(3) 1/1	0.5(8) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1.07(6) 1/1	1.41(5) 1/1
CF8	3.09(1) NA	3(2.5) 0/0	0(10.5) 1/1	1.4(7) 1/1	2.9(4) 1/1	3(2.5) 1/1	1.5(6) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1.33(8) 1/1	1.59(5) 1/1
CF9	3.11(1) NA	3(3) 0/0	0(10.5) 1/1	1.8(6) 1/1	3(3) 1/1	3(3) 1/1	1.5(7) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1.42(8) 1/1	1.9(5) 1/1
CF10	2.11(1) NA	2(2.5) 0/0	0(10.5) 1/1	1.2(5.5) 1/1	1.2(5.5) 1/1	2(2.5) 1/1	1.1(7) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1(8) 1/1	1.26(4) 1/1
CF11	4(1) NA	4(1) 0/0	0(10) 1/1	1.3(5) 1/1	2.9(4) 1/1	4(1) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	0.32(7) 1/1	0.68(6) 1/1
CF12	3.1(1) NA	3.1(1) 0/0	0(10.5) 1/1	1.7(5.5) 1/1	2.4(4) 1/1	2.84(3) 1/1	1.6(7) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1.25(8) 1/1	1.7(5.5) 1/1
CF13	4(1) NA	3.9(2) 0/0	0(10.5) 1/1	0.9(7) 1/1	2.4(4) 1/1	3.78(3) 1/1	0.3(8) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1.61(5) 1/1	1.39(6) 1/1
CF14	1.12(1) NA	1(5) 0/0	0(10.5) 1/1	1(5) 0/0	1(5) 0/0	1(5) 1/1	1(5) 0/0	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1(5) 1/1	1(5) 1/1
CF15	4.52(1) NA	4.4(2) 0/0	0(10.5) 1/1	1.6(7) 1/1	3.8(4) 1/1	4(3) 1/1	1.2(8) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	2.1(5) 1/1	2(6) 1/1
Total ranks	15	34.5	150.5	99.5	58	40	105.5	157.5	157.5	157.5	102	82.5

Table 6. Average number of global peaks found for test function set 2 (CF1-CF15) with D=30 and the respective ranks (in parentheses) , *t*-test and Wilcoxon test on the average number of peaks for test functions are shown in the second row of each cell, which is presented as "*t*-test / Wilcoxon test".

Fun	SSDL CDE	SDL CDE	CDE	SDE	CCDE	CSDE	FER-PSO	SPSO	r2pso	r3pso	CMA	SCMA
CF1	4(1) NA	4(1) 0/0	0(10) 1/1	1.25(4) 1/1	0.9(5) 1/1	1.8(3) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	0.32(6) 1/1	0.28(7) 1/1
CF2	2(1) NA	2(1) 0/0	1(8.5) 1/1	1(8.5) 1/1	1.3(5) 1/1	1.5(3) 1/1	1.34(4) 1/1	0(11) 1/1	0(11) 1/1	0(11) 1/1	1.25(6) 1/1	1.2(7) 1/1
CF3	4(1) NA	4(1) 0/0	0(10.5) 1/1	0.88(5) 1/1	2.6(4) 1/1	3(3) 1/1	0.35(7) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	0.32(8) 1/1	0.6(6) 1/1
CF4	2(1) NA	2(1) 0/0	0(8.5) 1/1	0(8.5) 1/1	0.8(3) 1/1	0.6(4) 1/1	0(8.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	0(8.5) 1/1
CF5	2(1) NA	2(1) 0/0	0(8.5) 1/1	0(8.5) 1/1	0.7(3.5) 1/1	0.7(3.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	0(8.5) 1/1
CF6	3(1) NA	3(1) 0/0	0(10) 1/1	1(4.5) 1/1	0.6(7) 1/1	1(4.5) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	1(4.5) 1/1	1(4.5) 1/1
CF7	2(1) NA	1.97(2) 0/0	0(9.5) 1/1	1(4.5) 1/1	0.8(6) 1/1	1.4(3) 1/1	0(9.5) 1/1	0(9.5) 1/1	0(9.5) 1/1	0(9.5) 1/1	0(9.5) 1/1	1(4.5) 1/1
CF8	3(1) NA	3(1) 0/0	0(10) 1/1	1(3.5) 1/1	0.2(7) 1/1	1(3.5) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	0.46(6) 1/1	0.54(5) 1/1
CF9	3.6(1) NA	3.6(1) 0/0	0(9) 1/1	0(9) 1/1	1(4) 1/1	1.1(3) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	0.6(5) 1/1	0(9) 1/1
CF10	1.23(1) NA	1.05(2) 1/1	0(9) 1/1	0(9) 1/1	0.7(5) 1/1	1(3.5) 1/1	0.4(6) 1/1	0(9.5) 1/1	0(9.5) 1/1	0(9.5) 1/1	0(9.5) 1/1	1(3.5) 1/1
CF11	2(1) NA	2(1) 0/0	0(9) 1/1	0(9) 1/1	1.1(4) 1/1	1.2(3) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	0.24(5) 1/1	0(9) 1/1
CF12	1.8(1) NA	1.8(1) 0/0	0(10) 1/1	1(5.5) 1/1	1(5.5) 1/1	1.5(3) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	0(10) 1/1	1(5.5) 1/1	1(5.5) 1/1
CF13	3(1) NA	3(1) 0/0	0(9) 1/1	0.2(5) 1/1	0.6(4) 1/1	0.8(3) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1
CF14	1.4(1) NA	1.19(2) 1/1	0(9) 1/1	0(9) 1/1	0.5(5) 1/1	0.6(4) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	0(9) 1/1	1(3) 1/1	0(9) 1/1
CF15	2.13(1) NA	2.03(2) 0/0	0(10.5) 1/1	1(6.5) 1/1	1.2(4) 1/1	1.6(3) 1/1	1(6.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	0(10.5) 1/1	1(6.5) 1/1	1(6.5) 1/1
Total ranks	15	19	141.5	100.5	72	50	126	144	144	144	100.5	102.5

To clearly present the comparison results of all the niching algorithms on all the test functions in terms of the average number of global peaks found, Figs. 2 and 3 give a simplified visual comparison based on average number of peaks found by 18 algorithms. Please note that the results of each problem in Figs 2 and 3 are normalized for the convenience of comparison. From the results, it can be seen that both proposed algorithms rank the top two among all compared niching algorithms.

Table 7. Average number of global peaks found for test function set 1 (F1-F14) and the respective ranks

(in parentheses) , *t*-test and Wilcoxon test on the average number of peaks for test functions are shown in the second row of each cell, which is presented as "*t*-test / Wilcoxon test".

Fun	SSDL CDE	SDL CDE	Self-CSDE	NShDE	NSDE	LIPS	r2psolhc	r3psolhc	IWO- δ - GSO
F1	1(1) NA	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	0.5(9) 1/1	0.56(8) 1/1	0.6(7) 1/1	1(1) 0/0
F2	1(1) NA	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	0.37(9) 1/1	0.44(8) 1/1	0.56(7) 1/1	1(1) 0/0
F3	2(1) NA	2(1) 0/0	2(1) 0/0	2(1) 0/0	2(1) 0/0	0.37(9) 1/1	0.48(8) 1/1	0.6(7) 1/1	2(1) 0/0
F4	5(1) NA	5(1) 0/0	5(1) 0/0	5(1) 0/0	5(1) 0/0	5(1) 0/0	5(1) 0/0	4.92(9) 0/0	5(1) 0/0
F5	1(1) NA	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0
F6	5(1) NA	5(1) 0/0	5(1) 0/0	5(1) 0/0	5(1) 0/0	5(1) 0/0	4.92(8) 0/0	4.88(9) 0/0	5(1) 0/0
F7	1(1) NA	1(1) 0/0	1(1) 0/0 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0	1(1) 0/0
F8	4(1) NA	4(1) 0/0	4(1) 0/0	3.92(6) 0/0	4(1) 0/0	4(1) 0/0	3(9) 1/1	3.12(8) 1/1	3.88(7) 1/1
F9	2(1) NA	2(1) 0/0	2(1) 0/0	2(1) 0/0	2(1) 0/0	2(1) 0/0	1.56(8) 1/1	1.48(9) 1/1	1.6(7) 1/1
F10	1(1) NA	1(1) 0/0	1(1) 0/0	0.96(7) 0/0	1(1) 0/0	1(1) 0/0	0.72(8) 1/1	0.6(9) 1/1	1(1) 0/0
F11	18(1) NA	18(1) 0/0	18(1) 0/0	18(1) 0/0	18(1) 0/0	17.82(6) 0/0	15.1(8) 1/1	16.2(7) 1/1	NA
F12	6(1) NA	6(1) 0/0	5.89(4) 0/0	5.88(5) 0/0	5.84(6) 0/0	4.93(9) 1/1	5.36(7) 1/1	5.28(8) 1/1	6(1) 0/0
F13	35.98(1) NA	35.96(2.5) 0/0	33(4) 1/1	35.96(2.5) 0/0	30.6(5) 1/1	21.26(9) 1/1	22.5(8) 1/1	23.1(7) 1/1	29.6(6) 1/1
F14	192(1) NA	190.4(2) 0/0	118.86(4) 1/1	179(3) 1/1	84.28(6) 1/1	61.07(7) 1/1	42.2(9) 1/1	43.3(8) 1/1	102.6(5) 1/1

Table 8. Average number of global peaks found for test function set 2 (CF1-CF15) with D=10 and the respective ranks (in parentheses) , t-test and Wilcoxon test on the average number of peaks for test functions are shown in the second row of each cell, which is presented as "t-test / Wilcoxon test".

Fun	SSDL CDE	SDL CDE	Self-CSDE	NShDE	NSDE	LIPS	r2psolhc	r3psolhc	IWO- δ - GSO
CF1	6.91(1) NA	6.88(3) 0/0	6.9(2) 0/0	3.7(5.5) 1/1	6.7(4) 1/1	3.7(5.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	2(7) 1/1
CF2	4.1(1) NA	4(3) 0/0	4(3) 0/0	2.8(5) 1/1	4(3) 0/0	2.1(6) 1/1	0(8.5) 1/1	0(8.5) 1/1	2(7) 1/1
CF3	6(1) NA	5.96(4) 0/0	6(1) 0/0	4(5.5) 1/1	6(1) 0/0	4(5.5) 1/1	0(8.5) 1/1	0(8.5) 1/1	3.6(7) 1/1
CF4	5.7(1) NA	5.6(2.5) 0/0	5.6(2.5) 0/0	4.5(5) 1/1	5.4(4) 1/1	1.9(6) 1/1	0(8) 1/1	0(8) 1/1	0(8) 1/1
CF5	6(1) NA	6(1) 0/0	6(1) 0/0	3.6(5) 1/1	5.9(4) 0/0	2.2(6) 1/1	0(8.5) 1/1	0(8.5) 1/1	2(7) 1/1
CF6	3.83(1) NA	3.71(2) 0/0	3(5) 1/1	3(5) 1/1	3(5) 1/1	3.7(3) 1/1	0(8.5) 1/1	0(8.5) 1/1	1.52(7) 1/1
CF7	3.37(1) NA	3.23(2) 0/0	1.98(3) 1/1	1(7) 1/1	1.9(4) 1/1	1.4(6) 1/1	0(8.5) 1/1	0(8.5) 1/1	1.8(5) 1/1
CF8	3.09(1) NA	3(4) 0/0	3(4) 0/0	3(4) 0/0	3(4) 0/0	3(4) 0/0	0(8.5) 1/1	0(8.5) 1/1	1.3(7) 1/1
CF9	3.11(1) NA	3(3.5) 0/0	3(3.5) 0/0	3(3.5) 0/0	3(3.5) 0/0	2.4(6) 1/1	0(8.5) 1/1	0(8.5) 1/1	1.8(7) 1/1
CF10	2.11(1) NA	2(3.5) 0/0	2(3.5) 0/0	1(6) 1/1	2(3.5) 0/0	2(3.5) 0/0	0(7.5) 1/1	0(7.5) 1/1	NA
CF11	4(1) NA	4(1) 0/0	4(1) 0/0	2.2(6) 1/1	4(1) 0/0	3.4(5) 1/1	0(7.5) 1/1	0(7.5) 1/1	NA
CF12	3.1(1) NA	3.1(1) 0/0	2.94(3) 1/1	2(6) 1/1	2.9(4) 1/1	2.6(5) 1/1	0(7.5) 1/1	0(7.5) 1/1	NA
CF13	4(1) NA	3.9(2.5) 0/0	3.9(2.5) 0/0	1(6) 1/1	3.8(4) 1/1	3.6(5) 1/1	0(7.5) 1/1	0(7.5) 1/1	NA
CF14	1.12(1) NA	1(4) 0/0	1(4) 0/0	1(4) 0/0	1(4) 0/0	1(4) 0/0	0(7.5) 1/1	0(7.5) 1/1	NA
CF15	4.52(1) NA	4.4(2) 0/0	4(3.5) 1/1	2.4(6) 1/1	4(3.5) 1/1	3.8(5) 1/1	0(7.5) 1/1	0(7.5) 1/1	NA

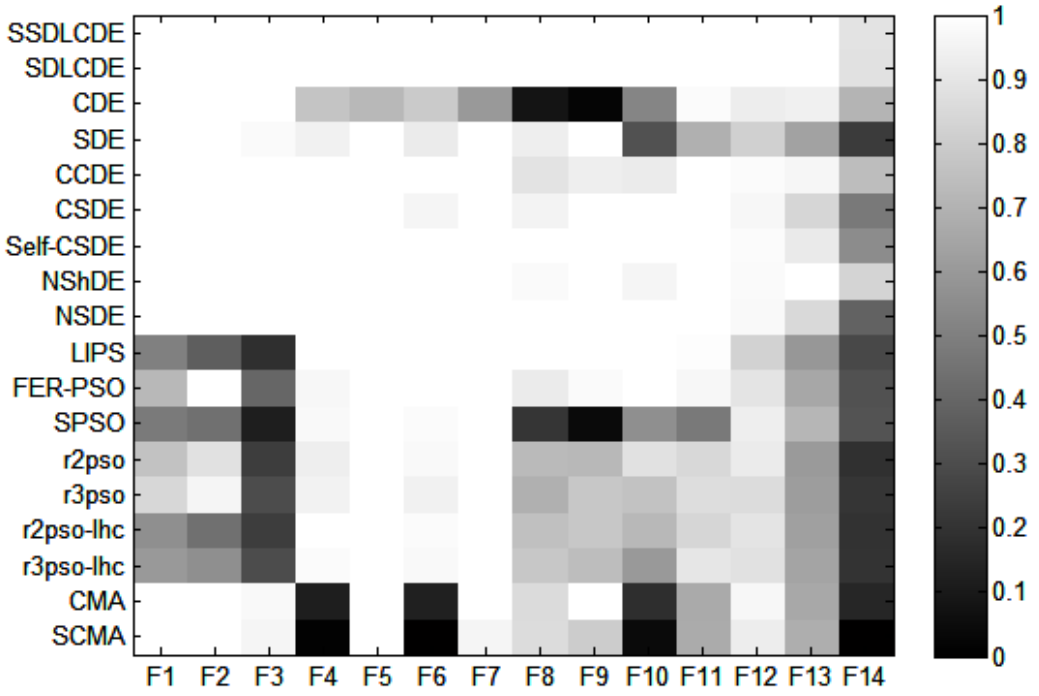

Fig.2. Overview of average number of global optima found by 18 niching algorithms on test functions set 1, where 1 (white) refers to the best while 0 (black) indicate the worst.

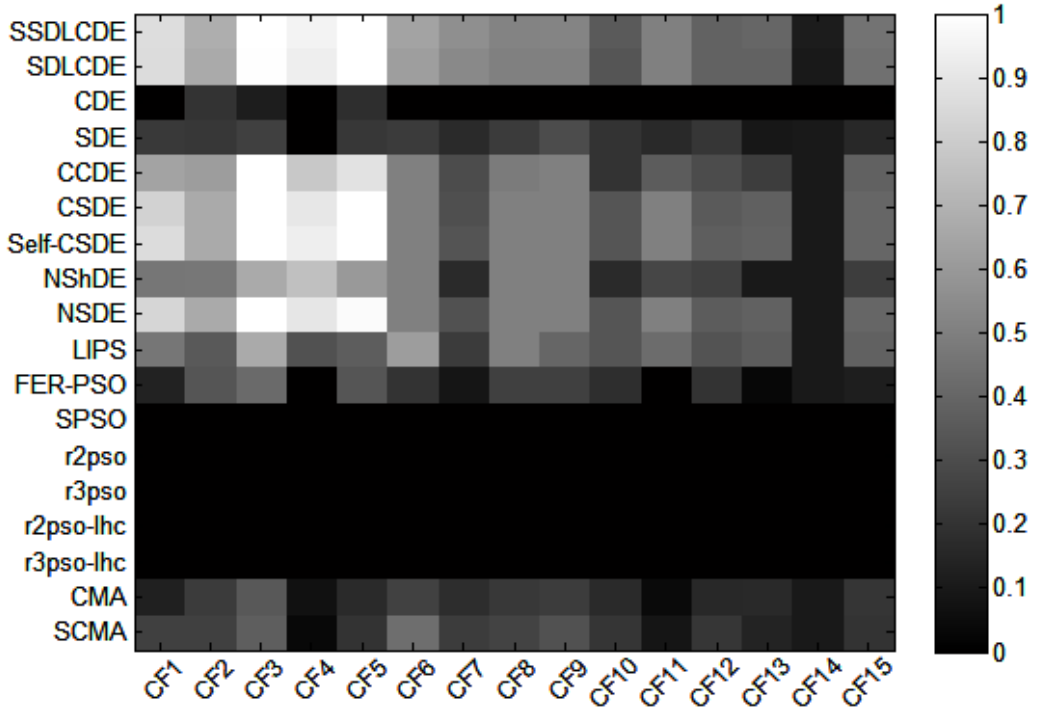

Fig.3. Overview of average number of global optima found by 18 niching algorithms on test functions set 2, where 1 (white) refers to the best while 0 (black) indicate the worst.

Further experiment results of 12 niching algorithms in terms of another two assessment criteria are summarized in Tables 9 and 10. Both the assessment criteria adopted are presented as follows.

1) Peak Accuracy

The peak accuracy indicates the absolute difference in function fitness between each true $peak_i$ (i = 1, 2, ... , #peaks) and its closest individual x in population. The level of peak accuracy reflects how close in fitness the true peaks and their corresponding most-fit individuals in the final population. If the fitness value of individual x is denoted by $f(x)$, the peak accuracy is calculated using the following equation:

$$\text{peak accuracy} = \sum_{i=1}^{\#peaks} \left| f(peak_i) - f(x_i) \right| \tag{19}$$

2) Distance Accuracy

The only peak accuracy is insufficient to indicate absolute validity, it may lead to erroneous results, because there are relatively flat fitness surfaces on the peaks. Under exceptional circumstances, Distance between true peaks and their closest individual x is needed as another indicator of true accuracy. The distance accuracy is computed the same way as peak accuracy, with the only change that the fitness values are replaced by the Euclidean distance, which is as below.

$$\begin{aligned}\text{distance accuracy} = \sum_{i=1}^{\#peaks} &(\text{Euclidean distance} \\ &\text{between } peak_i \text{ and individual } x)\end{aligned} \tag{20}$$

The compared results of each niching techniques on test set 1 are listed in Tables 9 and 10. The ranks of each method are presented in the parentheses of each cell. From both tables, it can be seen that our two proposed algorithms rank on the top two on most of 14 test functions in terms peak accuracy and Euclidean distance accuracy. It is clear that our proposed algorithms actually have overwhelming advantages over other algorithms in addition CCDE and CSDE. For only function 1 in test set 1, one of our algorithms (SDLCDE) ranks after the algorithm SDE in terms of distance accuracy. Nevertheless, SSDLCDE still ranks on the top. This also illustrates the effectiveness of self-adaptive strategy from another perspective.

Table 9. Peak accuracy of test function set 1 (F1-F14) and the respective ranks (in parentheses)

Fun	SSDL CDE	SDL CDE	CDE	SDE	CCDE	CSDE	FER-PSO	SPSO	r2pso	r3pso	r2psolhc	r3psolhc
F1	1.29e-5 (1)	2.03e-5 (2)	5.72e-2 (11)	4.63e-5 (4)	1.72e-4 (5)	4.02e-5 (3)	3.39e-2 (6)	4.36e-2 (9)	4.19e-2 (8)	4.15e-2 (7)	1.32e-1 (12)	5.47e-2 (10)
F2	1.13e-6 (1)	2.81e-6 (2)	4.41e-2 (9)	7.34e-5 (4)	1.75e-4 (5)	2.97e-6 (3)	1.04e-3 (6)	8.87e-2 (12)	1.22e-2 (7)	1.95e-2 (8)	5.64e-2 (11)	4.49e-2 (10)
F3	2.24e-3 (1)	5.47e-2 (2)	3.52e-1 (5)	5.33 (12)	1.87e-1 (3)	2.31 (10)	4.01e-1 (7)	2.87 (11)	3.92e-1 (6)	3.35e-1 (4)	7.27e-1 (9)	4.55e-1 (8)
F4	1.93e-13 (1)	1.02e-12 (2)	6.23e-6 (10)	6.55e-11 (3)	2.25e-8 (7)	5.81e-6 (9)	1.98e-8 (6)	8.54e-10 (4)	3.37e-9 (5)	2.49e-5 (11)	3.08e-6 (8)	8.00e-5 (12)
F5	4.62e-11 (1)	3.57e-10 (2)	7.21e-7 (12)	9.62e-10 (3)	6.39e-8 (9)	1.05e-7 (11)	6.83e-8 (10)	6.14e-8 (8)	9.15e-9 (4)	3.28e-8 (7)	3.19e-8 (6)	2.86e-8 (5)
F6	4.83e-10 (1)	5.35e-9 (2)	4.94e-6 (8)	3.83e-8 (4)	2.56e-5 (10)	4.88e-7 (7)	2.02e-8 (3)	7.26e-8 (6)	2.01e-5 (9)	5.26e-5 (11)	6.64e-8 (5)	9.98e-5 (12)
F7	1.60e-10 (1)	2.26e-10 (2)	3.74e-6 (11)	1.71e-3 (12)	1.43e-7 (10)	3.60e-8 (7)	4.44e-8 (8)	6.29e-8 (9)	4.07e-9 (4)	3.33e-9 (3)	5.55e-9 (5)	6.29e-9 (6)
F8	2.83e-7 (1)	4.20e-7 (2)	5.40e-2 (10)	7.84e-6 (3)	9.57e-5 (7)	2.57e-3 (8)	2.29e-5 (6)	8.29e-6 (4)	7.40e-2 (11)	3.96e-2 (9)	1.04e-5 (5)	1.19e-1 (12)
F9	3.52e-11 (2)	8.04e-12 (1)	2.57e-4 (12)	4.73e-9 (4)	6.47e-8 (9)	2.95e-10 (3)	9.23e-8 (10)	2.96e-8 (6)	4.14e-8 (8)	2.66e-8 (5)	1.19e-7 (11)	3.25e-8 (7)
F10	1.48e-10 (1)	6.63e-10 (2)	4.24e-7 (7)	6.41 (11)	8.10e-10 (3)	3.08e-7 (5)	4.16e-7 (6)	3.38e1 (12)	8.10e-2 (8)	3.91e-9 (4)	4.61e-1 (10)	2.64e-1 (9)
F11	4.18e-3 (1)	1.17e-2 (2)	1.72e-1 (6)	3.06e2 (12)	1.49e-2 (3)	3.11e-2 (4)	4.39e-2 (5)	8.95e1 (11)	1.38 (8)	2.63 (9)	6.02 (10)	3.03e-1 (7)
F12	1.07e-5 (1)	1.52e-5 (2)	1.01e-4 (5)	8.20e-5 (4)	3.51e-5 (3)	1.08e-4 (6)	9.02e-3 (12)	7.36e-3 (11)	3.80e-4 (9)	7.27e-4 (10)	2.82e-4 (8)	2.55e-4 (7)
F13	4.58e-5 (1)	5.82e-5 (2)	5.80e-4 (6)	2.96e-4 (5)	6.82e-5 (3)	1.84e-4 (4)	1.03e-2 (7)	2.48e-1 (12)	5.31e-2 (11)	5.26e-2 (10)	1.91e-2 (8)	2.00e-2 (9)
F14	1.28e-4 (1)	1.54e-4 (2)	6.96e-3 (6)	3.11e-4 (3)	9.87e-4 (4)	2.18e-3 (5)	1.42e-1 (7)	1.64e1 (12)	2.26 (11)	1.76 (10)	1.56 (9)	9.97e-1 (8)
Total ranks	15	27	118	84	81	85	99	127	109	108	117	122

D. Locating Local Optima

In the real world, it is needed to locate not only the global peeks but also the local peeks. So a satisfactory niching algorithm should have the ability to search global optima as well as the local optima. In order to test the ability of locating both global and local optima, six test functions (F1, F2, F3, F5, F7, F10) from set 1 are used. To evaluate the performance of each niching algorithm, here we also adopt two criteria which are success rate and average number of optima found, respectively. Tables 11 and 12 present the results. It can be seen that with the double-layer-clustering and the self-adaptive strategy the ability of exploration and

exploitation is greatly improved. In order to give a clearer view, the final population distribution of SSDLCDE is also plotted on test functions F2, F3, F7 and F10, which are shown in Fig. 4-7, respectively. Please note that the population size and maximum number of function evaluations are set to 100 and 40 000 for function F10. And the other parameter settings are the same as those used in the previous experiments.

Table 10. Distance accuracy of test function set 1 (F1-F14) and the respective ranks (in parentheses)

Fun	SSDL CDE	SDL CDE	CDE	SDE	CCDE	CSDE	FER-PSO	SPSO	r2pso	r3pso	r2psolhc	r3psolhc
F1	3.24e-7 (1)	5.09e-7 (3)	6.37e-2 (12)	4.81e-7 (2)	4.31e-6 (5)	1.87e-6 (4)	8.4e-4 8(6)	1.43e-3 (10)	1.04e-3 (8)	1.03e-3 2(7)	3.38e-3 (11)	1.36e-3 (9)
F2	2.45e-7 (1)	4.54e-7 (2)	1.19e-3 (10)	8.93e-7 (4)	4.38e-6 (5)	7.42e-7 (3)	2.62e-5 (6)	2.27e-3 (12)	3.06e-4 (7)	4.85e-4 (8)	1.46e-3 (11)	1.12 e-3 (9)
F3	2.81e-5 (1)	6.84e-4 (2)	4.42e-3 (5)	9.17e-1 (12)	2.34e-3 (3)	2.89e-2 (10)	5.02e-3 (7)	2.69e-1 (11)	4.95e-3 (6)	4.18e-3 (4)	9.32e-3 (9)	5.68e-3 (8)
F4	8.45e-10 (1)	8.78e-9 (2)	1.32e-4 (6)	8.75e-2 (11)	4.21 (12)	1.78e-5 (5)	6.67e-3 (7)	1.74e-7 (4)	4.00e-8 (3)	1.33e-2 (9)	7.14e-3 (8)	2.41e-2 (10)
F5	2.81e-9 (1)	5.3e-8 (2)	2.37e-5 (12)	5.95e-8 (3)	1.79e-6 (10)	2.45e-6 (11)	5.68e-7 (7)	3.17e-7 (4)	4.15e-7 (5)	7.59e-7 (8)	5.54e-7 (6)	8.64e-7 (9)
F6	3.08e-9 (1)	7.41e-8 (2)	1.15e-4 (8)	9.03e-2 (12)	3.07e-5 (7)	1.09e-5 (6)	1.84e-6 (5)	9.67e-7 (4)	3.33e-2 (11)	1.67e-2 (10)	3.44e-7 (3)	6.83e-4 (9)
F7	5.26e-9 (1)	8.35e-9 (2)	3.27e-5 (11)	5.55e-3 (12)	2.62e-6 (10)	3.17e-7 (9)	9.61e-8 (8)	1.33e-8 (5)	1.01e-8 (3)	7.66e-8 (6)	1.26e-8 (4)	9.21e-8 (7)
F8	6.61e-5 (1)	8.64e-5 (2)	6.97e-2 (10)	2.29 (12)	1.23e-3 (6)	4.31e-3 (7)	4.87e-4 (4)	1.82e-4 (3)	4.23e-2 (8)	6.07e-2 (9)	7.38e-4 (5)	4.01e-1 (11)
F9	1.64e-10 (2)	4.57e-11 (1)	4.16e-3 (12)	7.07e-8 (5)	3.67e-5 (11)	2.14e-9 (3)	1.36e-7 (6)	6.12e-8 (4)	4.26e-6 (9)	7.37e-6 (10)	1.56e-7 (7)	5.93e-7 (8)
F10	2.38e-9 (1)	2.65e-9 (2)	5.15e-2 (7)	3.39e1 (12)	6.84e-9 (3)	2.89e-2 (6)	1.45e-7 (4)	0.84e-1 (8)	1.07 (9)	1.96e-2 (5)	2.9 (11)	2.16 (10)
F11	5.56e-3 (2)	2.04e-2 (4)	3.13e-2 (5)	8.02 (12)	7.8e-3 1(3)	1.53e-3 (1)	9.14e-1 (11)	7.23e-1 (10)	7.03e-2 (8)	6.92e-2 (7)	1.19e-1 (9)	3.43e-2 (6)
F12	2.03e-4 (1)	4.09e-4 (2)	1.08e-3 (3)	2.20e-1 (6)	9.93e-3 (4)	7.78e-2 (5)	4.41e-1 (10)	3.37e-1 (7)	3.41e-1 (8)	8.46e-1 (12)	3.82e-1 (9)	6.33e-1 (11)
F13	3.5e-2 (1)	7.82e-2 (2)	2.70e-1 (4)	5.15 (6)	1.40e-1 (3)	5.95e-1 (5)	9.68 (11)	6.07 (7)	8.62 (9)	10.6 (12)	7.76 (8)	8.91 (10)
F14	4.48e-3 (1)	5.32e-3 (2)	1.28e1 (5)	1.69e2 (10)	6.97e-2 (4)	2.82e-2 (3)	1.75e2 (11)	1.03e2 (6)	1.47e2 (8)	1.80e2 (12)	1.36e2 (7)	1.51e2 (9)
Total ranks	16	30	110	119	86	78	103	95	102	119	108	126

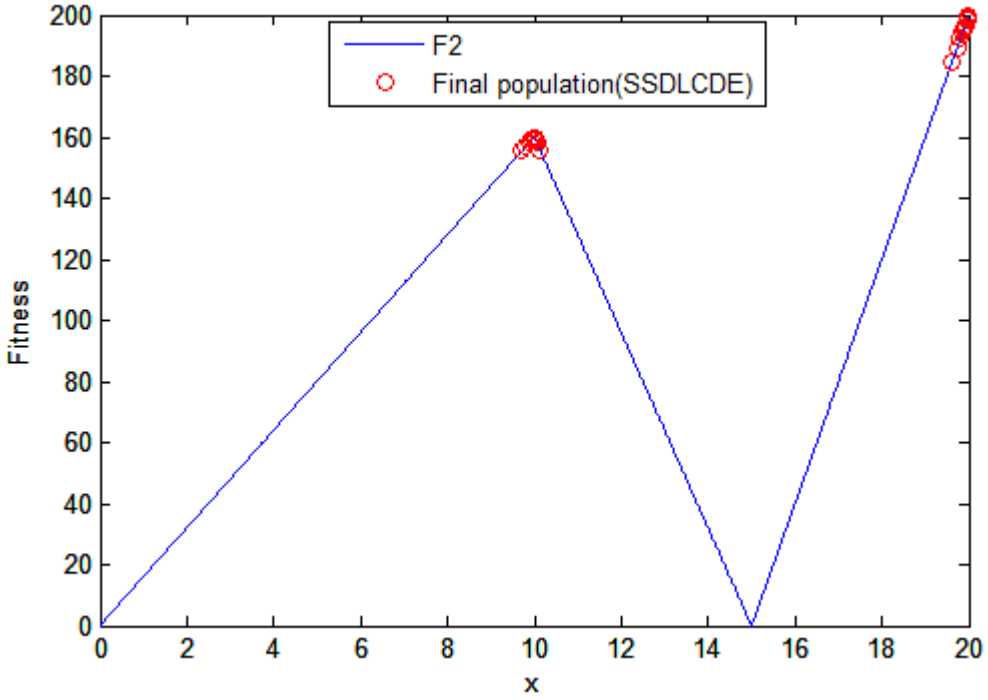

Fig.4. Final population of SSDLCDE for F2 in test set 1.

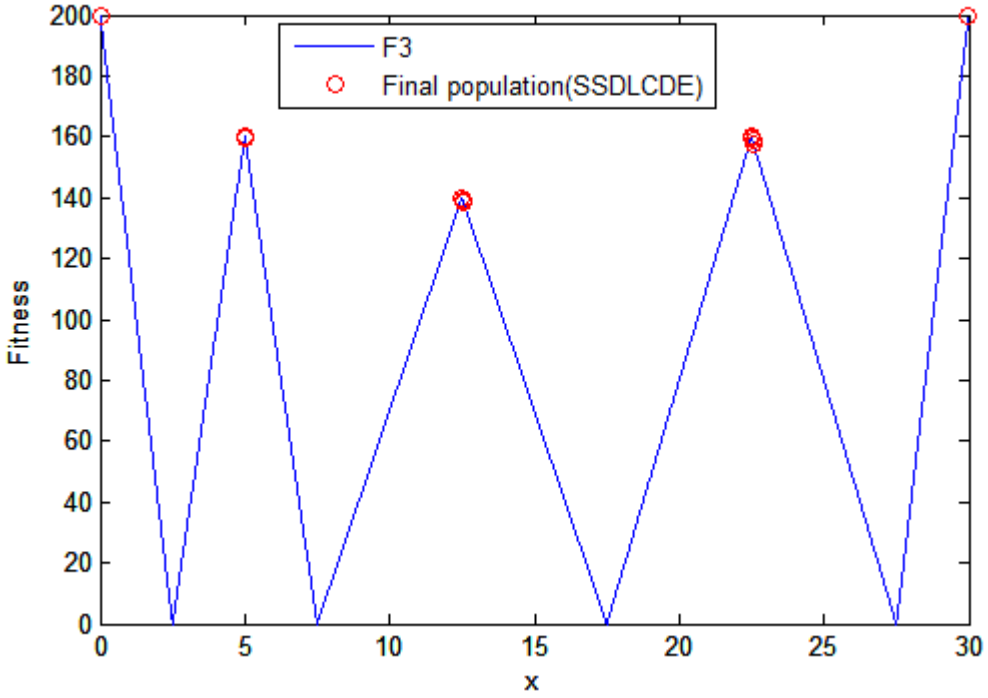

Fig.5. Final population of SSDLCDE for F3 in test set 1.

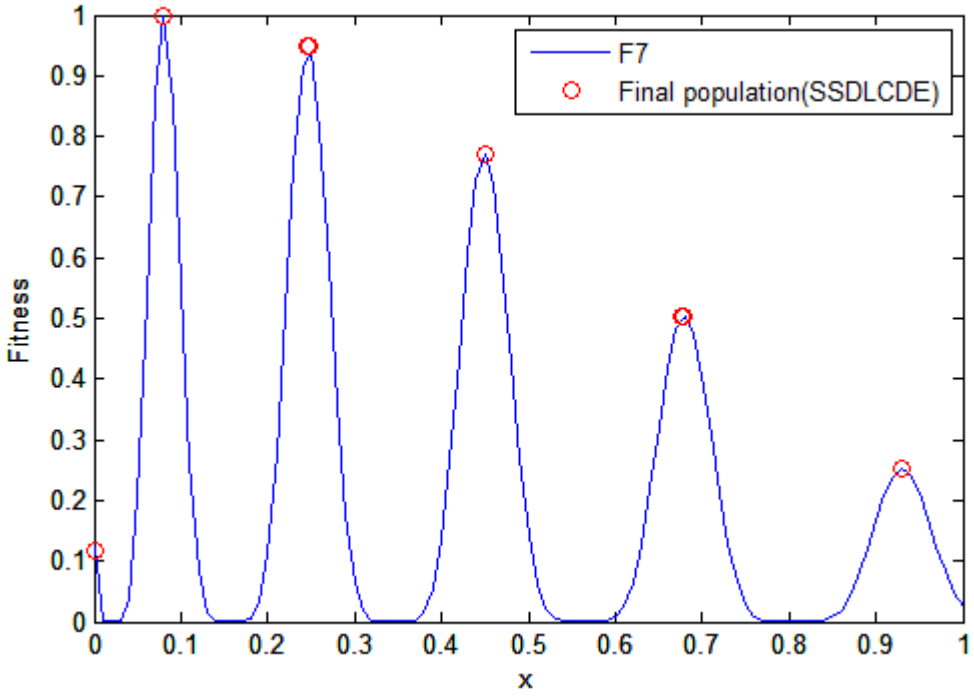

Fig.6. Final population of SSDLCDE for F7 in test set 1.

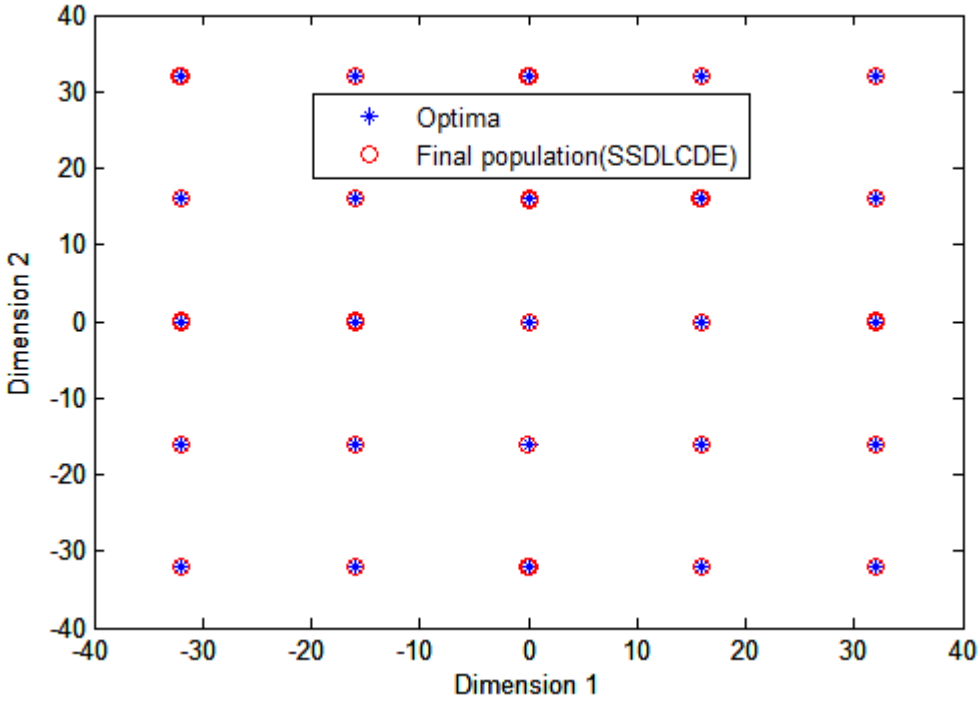

Fig.7. Final population of SSDLCDE for F10 in test set 1.

Table 11. Success rates (%) in locating both global and local peaks and the respective ranks (in parentheses)

Fun	SSDL CDE	SDL CDE	CDE	SDE	NSDE	CSDE	FER-PSO	SPSO	r2pso	r3pso	r2psolhc	r3psolhc
F1	100(1)	100(1)	100(1)	84(6)	100(1)	100(1)	64(8)	44(12)	72(7)	56(9)	48(11)	52(10)
F2	100(1)	100(1)	100(1)	68(9)	100(1)	100(1)	88(6)	72(8)	56(10)	32(12)	52(11)	76(7)
F3	100(1)	97(2)	44(3)	7(7)	40(5)	42(4)	0(10)	0(10)	0(10)	0(10)	8(6)	0(10)
F5	100(1)	100(1)	48(7)	0(10.5)	76(5)	88(4)	0(10.5)	100(1)	0(10.5)	0(10.5)	64(6)	4(8)
F7	100(1)	100(1)	3(8)	0(10.5)	87(5)	47(6)	0(10.5)	100(1)	0(10.5)	0(10.5)	96 (4)	40(7)
F10	100(1)	100(1)	0(11)	0(11)	100(1)	100(1)	0(11)	92(5)	60(8)	52(9)	84(6)	76(7)
Total ranks	6	7	31	54	18	17	56	37	56	61	44	49

Table 12. Average number of optima found in locating both global and local peaks and the respective ranks (in parentheses), t-test and Wilcoxon test on the average number of peaks for test functions are shown in the second row of each cell, which is presented as “t-test / Wilcoxon test”.

Fun	SSDL CDE	SDL CDE	CDE	SDE	CCDE	CSDE	FER-PSO	SPSO	r2pso	r3pso	CMA	SCMA
F1	2(1) NA	2(1) 0/0	2(1) 0/0	1.84(6) 1/1	2(1) 0/0	2(1) 0/0	1.48(10) 1/1	1.44(12) 1/1	1.72(7) 1/1	1.48(10) 1/1	1.48(10) 1/1	1.52(8) 1/1
F2	2(1) NA	2(1) 0/0	2(1) 0/0	1.68(9) 1/1	2(1) 0/0	2(1) 0/0	1.88(6) 1/1	1.72(8) 1/1	1.36(11) 1/1	1.24(12) 1/1	1.52(10) 1/1	1.76(7) 1/1
F3	5(1) NA	4.97(2) 0/0	4.44(3) 1/1	3.04(7) 1/1	3.76(5) 1/1	4.22(4) 1/1	0.64(11) 1/1	3.08(6) 1/1	0.8(10) 1/1	0.4(12) 1/1	3(8) 1/1	2.16(9) 1/1
F5	5(1) NA	5(1) 0/0	4.28(7) 1/1	1.52(9) 1/1	4.76(5) 0/0	4.86(4) 0/0	1(11) 1/1	5(1) 0/0	1(11) 1/1	1(11) 1/1	4.52(6)	2.8(8)
F7	5(1) NA	5(1) 0/0	1.56(12) 1/1	1.9(11) 1/1	4.87(5) 1/1	4.37(6) 1/1	1.97(10) 1/1	5(1) 0/0	2.53(8) 1/1	2.17(9) 1/1	4.97(4) 0/0	4.33(7) 1/1
F10	25(1) NA	25(1) 0/0	12.5(10) 1/1	1.32(12) 1/1	25(1) 0/0	25(1) 0/0	5.16(11) 1/1	24.9(5) 0/0	24.4(8) 0/0	24.3(9) 0/0	24.8(6) 0/0	24.6(7) 0/0
Total ranks	6	7	34	54	18	17	59	33	55	63	44	46

E. Maintaining the Optima Once found

In addition to having a great ability of locating global and local optima, a good niching algorithm must be able to maintain these optima throughout the entire process of searching, with respect to population size [10]. For a multimodal optimization technique, if it does not have a stable capability to maintain the optimal solution, even though some optima are

located during the search process, these optima will eventually lost. Both SSDLCDE and SDLCDE have the ability of maintaining the found optima until the end of the evolution. This is because both algorithms adopt the multi-population strategy based on double-layer-clustering. Once an individual moves around one global or local peak, the niche to which the individual belongs will continue to search better solutions near this peak. Only when a member detects a better solution within the niche, it is updated by the better solution. This search mechanism will maintain until the end of the run or the stop criteria is satisfied. The niching behavior of SSDLCDE on function F4 in test set 1 is given in Fig.8. From this figure, it can be observed that SSDLCDE is able to develop stable niches around the global optima.

F. Effect of varying the subpopulation size in the first layer clustering

Population size is a very important parameter for a niching algorithm. If the population size is too small, both exploration and exploitation ability of the niching algorithm will be greatly reduced, which leads to the failure of finding all the optima. On the other hand, if the population size is too large, the computation costs of the algorithm will be increased.
In addition to the population size, the subpopulation size is another significant parameter that affects the performance of the niching algorithm. In fact, each niche is a small population in which the individuals perform a basic optimization algorithm. In general, under the premise of a certain population size, a smaller subpopulation size will enhance the diversity of the population, thereby improving the exploration ability of the algorithm. However, this will also reduce the exploitation ability of the algorithm, which is not conducive to the convergence of the algorithm. On the contrary, a larger subpopulation size will increase the exploitation ability of the population and accelerate the convergence of the algorithm. However, it will also reduce the ability to exploit and globally search. Therefore, how to select a suitable subpopulation size depends on the characteristics of the function itself and the accuracy value set by the user.

In order to test the impact of varying the subpopulation size on SSDLCDE, all 14 problems in test function set 1 are investigated. The mean values of peak ratio (the percentage of successfully detected optima) within 30 runs are reported in Table 13. Please note, “NA” in the fifth column denotes that the population size NP can’t be divided by the selected subpopulation size M with no remainder, and “NA” in the sixth column represent that the number of species do not reach the minimum requirement of 4, because the experience tells us that the minimum population size of DE algorithms should be set to a number much higher than 4 [39]. The population size (NP) and the subpopulation size (M) adopted are also shown in this table, and the settings of other parameters are the same as the previous corresponding experiments. In this experiment, it can be observe that the represented results are basically consistent with our analysis above. For the test functions F1-F10 and F12, the peak ratio decreases as the subpopulation size increases in most cases. However, for the test functions F11, F13, and F14, when the subpopulation M is equal to 10, the peak ratio has achieved the maximum. This because these three are challenging functions with many optima and complex functional landscape, if the M is smaller, although the diversity of population is improved, it is not beneficial for convergence. Note again, it is based on the results of this experiment that

the values of the subpopulation size (M) in the previous experiments were determined.

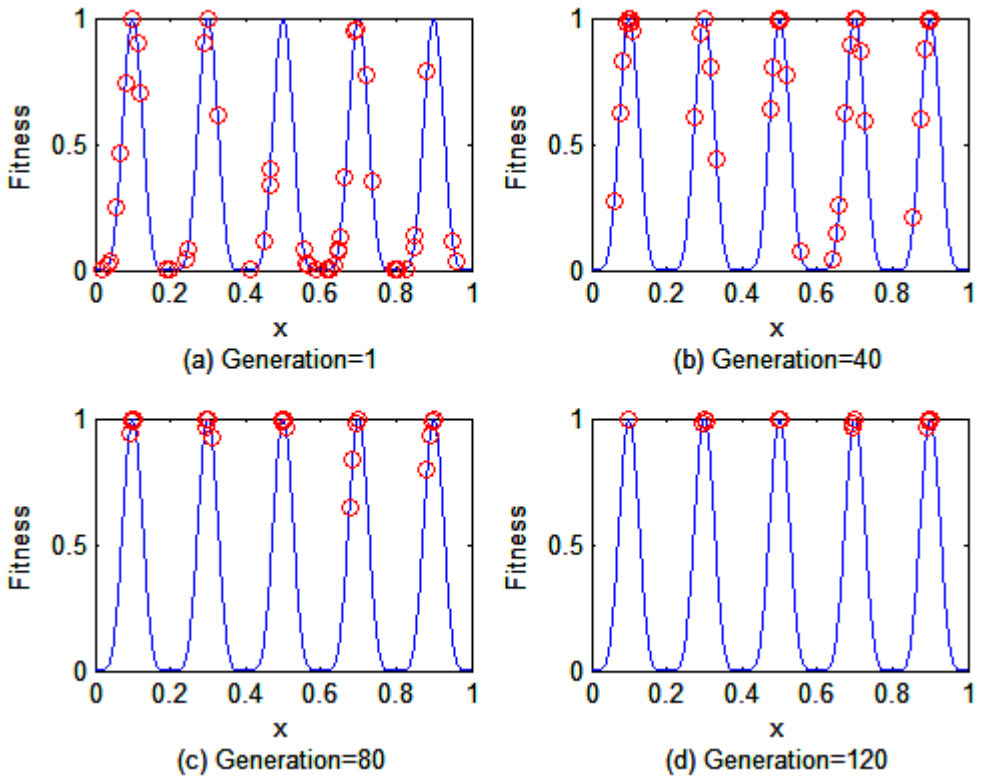

Fig.8. Distribution of population over function evaluations (F4 in test set 1).

6. Conclusion

Niching technique is an effective method to solve the multimodal problems. For most of niching algorithms, the whole population is first divided into multiple subpopulations, and then each niche separately searches the optima of the multimodal problem. In this way, these niching algorithms are able to perform well on the general multimodal optimization problems. However, for the complex problems with many peaks, there will be serious problem, that is, once the population has converged to some peaks found, it is difficult for the individuals to escape from these optima to search new peaks. In order to address this issue, this paper proposed a double-layer-clustering method based differential evolution using speciation and integrated it with self-adaptive strategy to solve multimodal optimization problems. The whole population is divided into numinous subpopulations by the first-layer-clustering. Subsequently, the double-layer-clustering method is used to drive the seeds of each species to search in the entire solution space. Compared with a single-layer-clustering method, the double-layer-clustering method ensures that the individuals are able to escape from local optima to search other optima and converge fast with a high accuracy. The results of experimental studies demonstrated that the proposed algorithms can outperform numerous state-of-the-art niching optimization algorithms on a large number of test functions. The experimental results also showed the effectiveness and the efficiency of the double-layer-

clustering strategy and self-adaptive strategy.

Since the double-layer-clustering can enhance the diversity of the population and increase the global search ability of the algorithm, how to apply this method to other niche algorithms will be studied in the future work. In addition, the future research may also focus on using more high-dimensional multimodal functions to test our algorithms.

Table 13 Effect of the varying subpopulation size (M) of the first layer clustering on SSDLCDE

Functions	Population Size (*NP*)	Subpopulation Size (*M*)			
		5	10	20	25
F1	50	1	1	NA	NA
F2	50	1	1	NA	NA
F3	50	1	0.83	NA	NA
F4	50	1	1	NA	NA
F5	50	1	1	NA	NA
F6	50	1	0.92	NA	NA
F7	50	1	0.93	NA	NA
F8	50	1	0.98	NA	NA
F9	50	1	0.91	NA	NA
F10	50	1	0.98	NA	NA
F11	250	0.99	1	NA	1
F12	100	1	0.98	0.9	0.97
F13	500	0.99	1	0.98	0.99
F14	1000	0.88	0.91	0.88	0.89

Acknowledgments

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Numbers 93539).

The authors certify that they have no conflict of interest regarding copyright and finance.

References:

[1] J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*: Morgan Kaufmann, 2001.

[2] X. Li, “Niching without niching parameters: particle swarm optimization using a ring topology,” *Evolutionary Computation, IEEE Transactions on,* vol. 14, no. 1, pp. 150-169, 2010.

[3] Z. Xu, Y. Wang, S. Li, Y. Liu, Y. Todo, and S. Gao, “Immune algorithm combined with estimation of distribution for traveling salesman problem,” *IEEJ Transactions on Electrical and Electronic Engineering,* vol. 11, no. S1, 2016.

[4] H. Wang, N. Zhang, and J.-C. Créput, “A massively parallel neural network approach to large-scale Euclidean traveling salesman problems,” *Neurocomputing,* vol. 240, pp. 137-151, 2017.

[5] R. Storn, and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of global optimization,* vol. 11, no. 4, pp. 341-359, 1997.

[6] R. C. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory." pp. 39-43.

[7] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization." pp. 1942-1948.

[8] D. Parrott, and X. Li, “Locating and tracking multiple dynamic optima by a particle swarm model using speciation,” *Evolutionary Computation, IEEE Transactions on,* vol. 10, no. 4, pp. 440-458, 2006.

[9] X. Li, "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization." pp. 105-116.

[10] S. W. Mahfoud, “Niching methods for genetic algorithms,” Urbana, 1995.

[11] J. Horn, “The nature of niching: Genetic algorithms and the evolution of optimal,” Cooperative Populations PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, IL, 1997.

[12] K. A. De Jong, “Analysis of the behavior of a class of genetic adaptive systems,” 1975.

[13] D. E. Goldberg, and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization." pp. 41-49.

[14] S. W. Mahfoud, “Crowding and preselection revisited,” *Urbana,* vol. 51, pp. 61801, 1992.

[15] D. Beasley, D. R. Bull, and R. R. Martin, “A sequential niche technique for multimodal function optimization,” *Evolutionary computation,* vol. 1, no. 2, pp. 101-125, 1993.

[16] X. Yin, and N. Germay, "A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization." pp. 450-457.

[17] G. R. Harik, "Finding Multimodal Solutions Using Restricted Tournament Selection." pp. 24-31.

[18] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms." pp. 798-803.

[19] M. Bessaou, A. Pétrowski, and P. Siarry, "Island model cooperating with speciation for multimodal optimization." pp. 437-446.

[20] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, “A species conserving genetic algorithm for multimodal function optimization,” *Evolutionary computation,* vol. 10, no. 3, pp. 207-234, 2002.

[21] R. Brits, A. P. Engelbrecht, and F. Van den Bergh, "A niching particle swarm optimizer." pp. 692-696.

[22] X. Li, "A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio." pp. 78-85.

[23] S. Bird, and X. Li, "Adaptively choosing niching parameters in a PSO." pp. 3-10.

[24] B. Qu, P. N. Suganthan, and J.-J. Liang, “Differential evolution with neighborhood mutation for multimodal optimization,” *IEEE transactions on evolutionary computation,* vol. 16, no. 5, pp. 601-614, 2012.

[25] B. Qu, P. N. Suganthan, and S. Das, “A distance-based locally informed particle swarm model for multimodal optimization,” *IEEE Transactions on Evolutionary Computation,* vol. 17, no. 3, pp. 387-402, 2013.

[26] W. Gao, G. G. Yen, and S. Liu, “A cluster-based differential evolution with self-adaptive strategy for

multimodal optimization," *IEEE transactions on cybernetics,* vol. 44, no. 8, pp. 1314-1327, 2014.

[27] R. Storn, and K. Price, *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*: ICSI Berkeley, 1995.

[28] R. Storn, "On the usage of differential evolution for function optimization." pp. 519-523.

[29] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization." pp. 17-24.

[30] A. W. Iorio, and X. Li, "Solving rotated multi-objective optimization problems using differential evolution." pp. 861-872.

[31] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE transactions on Evolutionary Computation,* vol. 13, no. 2, pp. 398-417, 2009.

[32] A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization." pp. 1785-1791.

[33] W. Gong, and Z. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Transactions on Cybernetics,* vol. 43, no. 6, pp. 2066-2081, 2013.

[34] J. Zhang, and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on evolutionary computation,* vol. 13, no. 5, pp. 945-958, 2009.

[35] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Information Sciences,* vol. 329, pp. 329-345, 2016.

[36] E. Yu, and P. N. Suganthan, "Ensemble of niching algorithms," *Information Sciences,* vol. 180, no. 15, pp. 2815-2833, 2010.

[37] S. Das, S. Maity, B.-Y. Qu, and P. N. Suganthan, "Real-parameter evolutionary multimodal optimization—A survey of the state-of-the-art," *Swarm and Evolutionary Computation,* vol. 1, no. 2, pp. 71-88, 2011.

[38] S. Hui, and P. N. Suganthan, "Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization," *IEEE transactions on cybernetics,* vol. 46, no. 1, pp. 64-74, 2016.

[39] X. Li, "Efficient differential evolution using speciation for multimodal function optimization." pp. 873-880.

[40] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*: Springer Science & Business Media, 2006.

[41] Y.-J. Wang, J.-S. Zhang, and G.-Y. Zhang, "A dynamic clustering based differential evolution algorithm for global optimization," *European Journal of Operational Research,* vol. 183, no. 1, pp. 56-73, 2007.

[42] R. Mallipeddi, and P. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies," *Swarm, evolutionary, and memetic computing,* pp. 71-78, 2010.

[43] Z. Cai, W. Gong, C. X. Ling, and H. Zhang, "A clustering-based differential evolution for global optimization," *Applied Soft Computing,* vol. 11, no. 1, pp. 1363-1379, 2011.

[44] R. Mallipeddi, P. N. Suganthan, Q.-K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing,* vol. 11, no. 2, pp. 1679-1696, 2011.

[45] S. Hui, and P. N. Suganthan, "Ensemble differential evolution with dynamic subpopulations and adaptive clearing for solving dynamic optimization problems." pp. 1-8.

[46] U. Halder, S. Das, and D. Maity, “A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments,” *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 881-897, 2013.

[47] G. Liu, and Z. Guo, “A clustering-based differential evolution with random-based sampling and Gaussian sampling,” *Neurocomputing*, vol. 205, pp. 229-246, 2016.

[48] R. Thomsen, "Multimodal optimization using crowding-based differential evolution." pp. 1382-1389.

[49] V. Kenneth, "Price, An introduction to differential evolution, New ideas in optimization," McGraw-Hill Ltd., UK, Maidenhead, UK, 1999.

[50] O. M. Shir, M. Emmerich, and T. Bäck, “Adaptive niche radii and niche shapes approaches for niching with the CMA-ES,” *Evolutionary Computation*, vol. 18, no. 1, pp. 97-126, 2010.

[51] S. Roy, S. M. Islam, S. Das, and S. Ghosh, “Multimodal optimization by artificial weed colonies enhanced with localized group search optimizers,” *Applied Soft Computing*, vol. 13, no. 1, pp. 27-46, 2013.

[52] B.-Y. Qu, and P. N. Suganthan, "Novel multimodal problems and differential evolution with ensemble of restricted tournament selection." pp. 1-7.

[53] D. H. Ackley, “An empirical study of bit vector function optimization,” *Genetic algorithms and simulated annealing*, vol. 1, pp. 170-204, 1987.

[54] K. Deb, *Genetic algorithms in multimodal function optimization*: Clearinghouse for Genetic Algorithms, Department of Engineering Mechanics, University of Alabama, 1989.

[55] Z. Michalewicz, *Genetic Algoriths + Data Structures = Evolution Programs*, New York: Springer-Verlag, 1996.

[56] K. DeJong, “An analysis of the behavior of a class of genetic adaptive systems,” *Ph. D. Thesis, University of Michigan*, 1975.

[57] O. M. Shir, and T. Bäck, "Niche radius adaptation in the cma-es niching algorithm," *Parallel Problem Solving from Nature-PPSN IX*, pp. 142-151: Springer, 2006.

[58] J. Gan, and K. Warwick, "A variable radius niching technique for speciation in genetic algorithms." pp. 96-103.